

**LIFELINES**

# The Software Magazine

\$3.00

December 1982

Volume III, No. 7 (ISSN 0279-2575, USPS 597-830)



# TIM<sup>TM</sup> III

## The Non-Programming Approach to Data Base Management

### Data Base Management

Data management packages were created to save time and money in the development of software solutions to information problems. Many have been designed to accomplish just that, although most have only the programmer in mind. Sure they would save time in the long run, but what of the initial investment in time and effort required to learn the new language? What about the non-programmers in the world who would like an easy yet powerful applications generator? The solution is one of the most highly acclaimed software packages of our time, T.I.M. III.

### What is T.I.M.?

T.I.M. is **Total Information Management**. Programmers love it due to its original solutions to classic data management problems. Non-programmers adore it since they can use it to achieve the same results as with other more complicated programming-like packages.

### What Makes T.I.M. So Simple to Use?

We at Innovative Software, Inc. designed T.I.M. from day one with the end user in mind. Maybe he is a programmer who doesn't have time to learn a new language. Or perhaps a neophyte who fears coding pads and lines numbered by tens. We felt that a data management package should be able to be used by anyone from a systems analyst to a secretary. That's why T.I.M. takes a full *menu-driven* approach, uses multiple *HELP* screens, and has a manual that sets a new standard in documentation.

### The Manual

Many people believe that the manual is just as important as the software itself, a view that we at Innovative Software, Inc. tend to share. The manual for T.I.M. is divided into two sections, the Reference section and the Primer. The Reference section describes all of T.I.M.'s commands and subcommands. This is done in English, not in technical terms or in our own language. Even if you have

never seen a computer before in your life, you'll be able to read and understand our manual immediately. The second section is a primer which goes through several examples for you, again in plain English. These true-to-life examples take the beginner by the hand, and instructs him what to do and when. You will be able to see for yourself that T.I.M.'s only limitation is the imagination of the user.

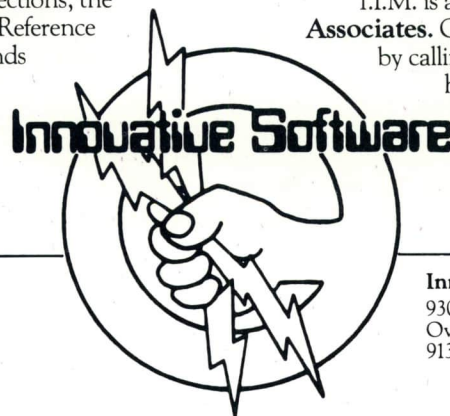
### Features of T.I.M.

T.I.M. has all of the features one has come to expect from a data management package, as well as many new ones. For example, a *word processing* interface that allows you to merge information from a T.I.M. file with letters or other documents created by a word processor. Now you can automatically send personalized letters to hundreds or thousands—quickly and easily. T.I.M.'s *Select* command enables you to pull specific information from a file. For example, "All customers who live in a certain ZIP code, whose last name begins with the letter A to L, whose balance due is less than \$50.00." A sophisticated *report generator* and even a *list generator* are also included.

How powerful is T.I.M.? With a maximum record size of 2400 characters and the ability to keep up to forty fields sorted properly at all times, T.I.M. is powerful enough to handle just about any application. T.I.M. can handle over 32,000 records per file, and two files can be linked together for reports if your application requires a many-to-one relationship. T.I.M. also includes all of the same editing commands as your word processor, thus making data entry and editing a snap. You can also pull selected records from one file to place them into another. Files may be restructured to add or subtract fields and/or change field lengths or types. T.I.M. even has its own utility for backing up hard disks onto floppies.

### Where to Find T.I.M.

T.I.M. is available from **Lifeboat Associates**. Or you may purchase from us direct by calling 913/383-1089. Either way you will have the finest data management program available.



Available for CP/M,\* and IBM PC DOS.\*\*  
CPM version—\$695. IBM PC version—\$495.

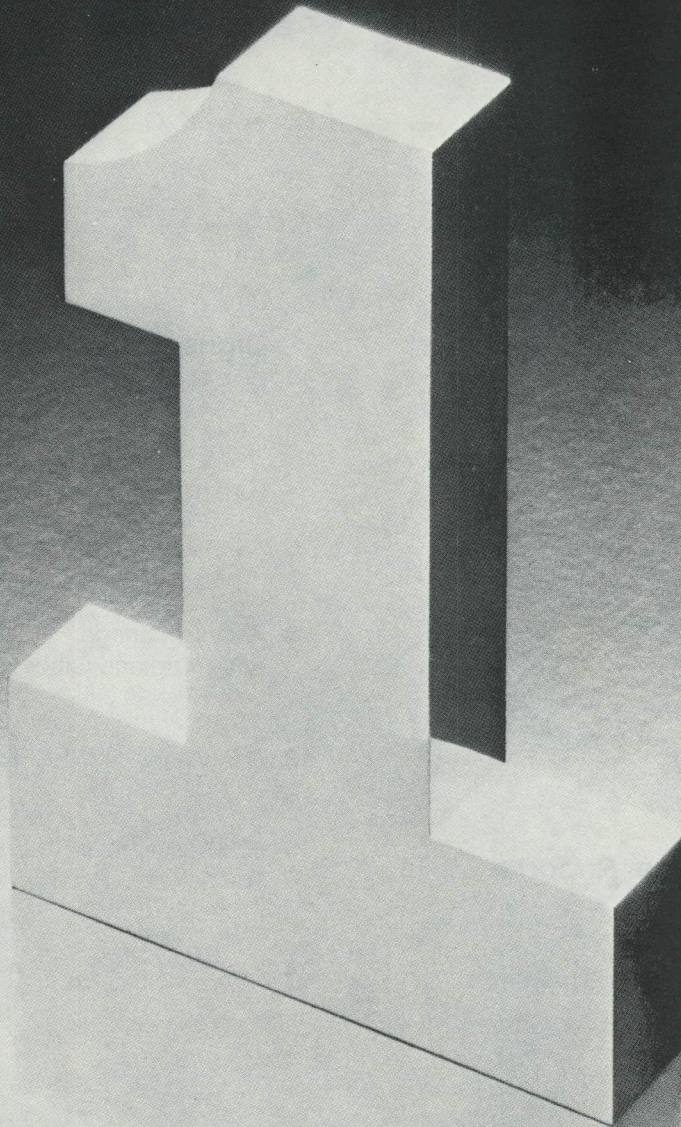
**Innovative Software, Inc.**  
9300 W. 110th Street, Suite 380  
Overland Park, Kansas 66210 USA  
913/383-1089

TIM is a Trademark of Innovative Software, Inc.  
\*CP/M and MP/M are Trademarks of Digital Research  
\*\*Trademarks of IBM

We are dedicated to the achievement of a singular goal . . .  
to market fully supported software that sets standards of excellence.

Standards against which all software will be measured.  
Standards which require that we, as well as the OEM's and authors  
with whom we labor, constantly offer the state-of-the-art.

Our commitment to being Number One is too strong for  
Lifeboat to market anything less.



## **Lifeboat Associates**

The standard for fully supported software.

1651 Third Avenue, NY, NY 10028. (212) 860-0300.  
TWX 710-581-2524 (LBSOFT NYK).

# LIFELINES

# The Software Magazine

December 1982

Volume III, No. 7

**Editor-in-Chief:** Edward H. Currie  
**Circulation/Customer Service:** Patricia Matthews  
**Design/Production:** K. Gartner  
**Typographer:** Harold Black

**Communications:** Bonita E. Taylor  
**Data Processing:** Lewis Tseng  
Lawrence Fishman  
Anne Odden

**Cover** by K. Gartner

## The CP/M® Users Group

- 32 CPMUG™ News  
Ward Christensen

## Software Notes

- 7 A Simple Menu Program  
Mikki D. Parkyn
- 34 Once More With Feeling  
CP/M Version 3  
Michael Olfe
- 36 New Versions of dBASE II™  
Michael Olfe

## Product Status Reports

- 30 New Products
- 31 New Versions
- 31 Books

## Digital Dollars Department

- 27 MicroMoneymaker's Forum  
Charles E. Sherman

## Features

- 5 A Review of Citation™  
Paul E. Hoffman
- 8 Access Manager™:  
Searching and Other Necessities  
Bruce H. Hunter
- 13 A Review of SUPERFILE™  
Bob Kowitt
- 18 8080 Assembler Programming  
Tutorial, Debugging  
Ward Christensen
- 24 RMST™, A Review  
Davis A. Foulger

**A PROFESSIONAL SYSTEM  
AT A P.C. PRICE**

**\$2995**

**TURN-KEY S-100 SYSTEM**

**FEATURING:**

Integrand 10 slot enclosure  
2 8" D.D., D.S. Drives  
ADDS 3A Viewpoint Terminal

Teletek Systemaster SBC  
2 Parallel & Serial Ports  
CPM™ 2.2 Installed

Full Teletek line available. Multi-user & Turbodos™ options can be added. Other S-100 products, printers, peripherals, personal computers, and CPM™ software products available at 15-20% above wholesale cost. Full service and repair. Workshops and classes held regularly.

**TOTAL ACCESS**  
SUITE 202, 2054 University Ave.  
Berkeley, California 94704  
415-652-3330 ext. 346

**FORTH-79**

**Version 2 For Z-80, CP/M (1.4 & 2.x),  
& NorthStar DOS Users**

The complete professional software system, that meets ALL provisions of the FORTH-79 Standard (adopted Oct. 1980). Compare the many advanced features of FORTH-79 with the FORTH you are now using, or plan to buy!

FEATURES	OURS	OTHERS
79-Standard system gives source portability.	YES	_____
Professionally written tutorial & user manual. 200 PG.	YES	_____
Screen editor with user-definable controls.	YES	_____
Macro-assembler with local labels.	YES	_____
Virtual memory.	YES	_____
BDOS, BIOS & console control functions (CP/M).	YES	_____
FORTH screen files use standard resident file format.	YES	_____
Double-number Standard & String extensions.	YES	_____
Upper/lower case keyboard input.	YES	_____
APPLE II/II+ version also available.	YES	_____
Affordable!	\$99.95	_____
Low cost enhancement options:		
Floating-point mathematics	YES	_____
Tutorial reference manual		
50 functions (AM9511 compatible format)		
Hi-Res turtle-graphics (NoStar Adv. only)	YES	_____
FORTH-79 V.2		\$99.95
ENHANCEMENT PACKAGE FOR V.2:		
Floating point		\$ 49.95
COMBINATION PACKAGE (Base & Floating point)		\$139.95
(advantage users add \$49.95 for Hi-Res)		
(CA. res. add 6% tax; COD & dealer inquiries welcome)		

**MicroMotion**

12077 Wilshire Blvd. # 506  
L.A., CA 90025 (213) 821-4340  
Specify APPLE, CP/M or Northstar  
Dealer inquiries invited.



**Make life easier with...**

**WASH™**

the latest in an easy to use CP/M directory maintenance utility that replaces a dozen older programs with its menu driven file handling capabilities like:

- DIRECTORY DISPLAY
- FILE VIEW AND PRINT
- FILE RENAME
- FILE DELETE
- FILE COPY
- FILE AND DISK STATISTICS
- TAGGED GROUP COPY AND DELETE

WASH is fully compatible with CP/M user area directories and is delivered with a user friendly installation program to adapt WASH to your 24 x 80 console. WASH is available on 8" single density diskette for \$49.95 plus 6% sales tax.



Contact:  
**MICRO RESOURCES**  
2468 Hansen Ct.  
Simi Valley, CA 93065  
(805) 527-7922

CP/M is a trademark of Digital Research

Copyright © 1982, by Lifelines Publishing Corporation. No portion of this publication may be reproduced without the written permission of the publisher. The single issue price is \$3.00 for copies sent to destinations in the U.S., Canada, or Mexico. The single issue price for copies sent to all other countries is \$4.30. All checks should be made payable to Lifelines Publishing Corporation. Foreign checks must be in U.S. dollars, drawn on a U.S. bank; checks, money orders, VISA, and MasterCard are acceptable. All orders must be pre-paid. Please send all correspondence to the Publisher at the below address.

Lifelines (ISSN 0279-2575, USPS 597-830) is published monthly at a subscription price of \$24 for twelve issues, when destined for the U.S., Canada, or Mexico, \$50 when destined for any other country. Second-class postage paid at New York, New York. POSTMASTER, please send changes of address to Lifelines Publishing Corporation, 1651 Third Ave., New York, N.Y. 10028.

Lifelines - TM Lifelines Publishing Corp.  
The Software Magazine - TM Lifelines Publishing Corp.  
SB-80, SB-86 - TMs Lifeboat Associates.  
BASIC-80, MBASIC, MS, SoftCard, COBOL-80 - TMs Microsoft, Inc.  
CB80, PL/I-80, SID-86, CP/M-86, Pascal MT +, MP/M, Access Manager - TMs, CP/M and CBASIC2 registered TMs - Digital Research, Inc.  
Citation - TM Eagle Enterprises  
dBASE II - TM Ashton-Tate.  
KIBITS - TM Bess Garber  
MailMerge, WordStar - TMs MicroPro International Corp.  
PMATE, PLINK-II - TMs Phoenix Software Associates, Ltd.  
RMS - TM Washington Computer Services  
SUPERFILE - TM FYI, Inc.  
Z80 - TM Zilog Corporation.  
Program names are generally TMs of their authors or owners.  
The CP/M Users Group is not affiliated with Digital Research, Inc.

### **It was inevitable.**

In the beginning, there was the data base management system. Powerful, but only if you knew programming. Then came the program generator—anyone could use it, but why bother to generate poorly written BASIC programs? Now there's the best of both worlds with QUICKCODE™, the data base program generator.

### **Power and ease of use.**

Fox & Geller's QUICKCODE™ combines power and ease of use in one neat package. It writes concise dBASE II™ programs to set up and maintain any kind of database. You can run them as is or customize them in seconds. And you'll still have all the power of dBASE II™ at your disposal: query language, report generator, and so on. But just as important: you don't need to do any programming. Just draw your data entry form on the screen and you're in business. Typical time to set up a customer list or order file: 5 minutes.

### **The Wordstar connection.**

QUICKCODE™ also gives you the ability to transfer your dBASE II™ data into Wordstar®/Mailmerge™ files for word processing and form letters. So you can get the most from two software bestsellers: dBASE II™ and Wordstar®.

(Software dealers: DOUBLE YOUR SALES!)

### **That's not all . . .**

There are lots of other features, like form and report generation up to 132 characters wide, four-up mailing labels, three kinds of data validation, four new data types not found in dBASE II™ itself, data base keys, and menu generators. You really have to see it to believe it.

### **It's your move.**

Now it's up to you to take advantage of this latest development in software. Why waste any more time writing programs or paying someone to write them for you?

Fox & Geller's QUICKCODE™: \$295.00.

QUICKCODE is now available for the IBM -PC with the Xedex Baby Blue Card.

---

# **QUICKCODE™**

## **The dBASE II™ Program Generator**

### **FOX&GELLER**

Fox & Geller, Inc.  
P.O. Box 1053  
Teaneck, NJ 07666  
201-837-0142

QUICKCODE is a trademark of Fox & Geller, Inc.  
dBASE II is a trademark of Ashton-Tate.  
WORDSTAR is a registered trademark of MicroPro International,  
San Rafael, California USA.  
MAILMERGE is a trademark of MicroPro International,  
San Rafael, California USA.  
IBM is a registered trademark of International Business Machines.

# A Review of Citation

Citation, version 2.0  
Eagle Enterprises  
2375 Bush Street  
San Francisco, CA 94115  
415-346-1249

Citation is an excellent utility for keeping track of articles, books, and advertisements. A pre-packaged database application, Citation is designed for anyone who needs to catalog information from many different sources. Although the concept behind it sounds simplistic, this is one of the best designed products of its kind that I have seen.

Many *Lifelines* readers are computer professionals who read at least three computer magazines a month. Most of us at some point have said to ourselves, "Gee, I should start cataloging all the articles that I read, in case I need to find one in the future." We often feel this way when we need to find an article and spend the better part of an hour searching through stacks of back issues. (Editor's Note: Of course an index is available for *Lifelines/The Software Magazine*.)

If Citation only catalogued, it would not be a very interesting program. Most people have data base management systems that could create an application for storing article listings, possibly with a keyword for each. But Citation also stores book references and name-and-address references, and gives you five words to index for each entry. Extremely easy to use, it supplies you with most of the features you would have built in yourself if you had the time and programming knowledge.

## Uses for Citation

Once you have Citation up and running (the installation is exceedingly easy), you will find yourself indexing more than you expected to. For example, I thought that I would only use it for indexing a few articles from *BYTE* and *Dr. Dobbs*, but I ended up using Citation on all the other computer

magazines I get as well. This happened when I realized that it takes only about five minutes per issue to index all the articles of possible interest - and the time I invested was quite little, compared to the tedium of digging through the magazines.

You will probably find yourself indexing more articles in a particular issue than you expected, even some you did not bother to read. (They may be needed in the future!) Since diskette space is inexpensive, it makes sense to include almost everything in sight before you put the magazine on the shelf, rather than deciding later that you want to look for articles on a subject you didn't index.

I also started using Citation to keep track of the ads in *BYTE*, so that I will be able to find new products and their manufacturers when I need them later. Citation's name-and-address feature is perfect for this, since these records are stored in the same files as the data on articles. Thus, when I look at all the citations for "hard disks," I can find the advertisements as well as the articles. Also, I keep a running list of the reviews in *Lifelines* and *InfoWorld*.

You can, of course, use Citation for other types of indexing. For instance, students writing dissertations and theses which involve many references will find Citation useful for collecting information and determining the usefulness of individual sources. It is certainly handy when a lengthy bibliography is required, although you will have to transcribe the information from Citation's output to proper bibliography format.

Citation is quite useful in cataloging public-domain software diskettes. Although most CPMUG and SIG/M diskettes have catalogs on them, you may want to keep track of programs of interest to you with Citation - in case you ever become interested in a particular class of programs (i.e., BDS C utilities, games, etc.) Since you have up to

**Paul E. Hoffman**

five keys to index each program or diskette, you can save the type of program, the language (in case you ever buy the compiler, you'll know what is currently available), or even the author.

There are many other applications for which Citation is well suited, limited only by your imagination. This review will discuss Citation as a computer literature indexing system, since I think that is what most *Lifelines* readers will use it for.

## Citation's Features

The most important considerations in judging a non-configurable application product with a specialized purpose are:

- whether you can do nearly everything that you want to do,
- and whether you can operate the system easily, so that you never hesitate to run it.

Citation has both of these features.

The features most important for an article indexing program are the ability to add and change records easily and great flexibility in searching for information. Citation covers the first feature by careful handholding throughout the program (without the condescending posture of some other turnkey programs). A display at the bottom of the screen is always present, telling you what you are doing. With five keys of twenty characters each, you will have no problems later on finding the record that you want.

Each citation record has the following fields:

Keywords	5 entries, 20 characters each
Source	25 characters
Date	in mm/dd/yy format
Page	4 digits
Summary	70 characters
Comments	9 lines of 70 characters

The source is the name of the magazine or periodical you are citing. The sum-

(continued next page)

mary is typed out in a keyword index report, but you can also use it simply as the first line of a ten line comment. There is plenty of space to record important future information about the article.

The publication record (for books or pamphlets) is the same as the citation record, except that you have a 20 character author field instead of a page field. The name-and-address screen has the same keywords, summary and comments fields, but also name, two address lines, city, state zip, and phone number. All three types of records are kept in the same data file. You can flip through a magazine page by page, adding articles or advertisements without changing data sets.

Citation is fully driven by menus, and a novice computer user should have absolutely no problem using it. All menus are clear and concise, and all have very good error handling and error messages. There is little that you can do wrong, and the escape key is used as a consistent way of getting out of the program at almost all levels.

Like most application programs, Citation uses (and expects) cursor addressing for all its functions. The program checks each character you type, and gives you many advanced editing features, such as automatic word wrap when entering in the comments field. The terminal installation program is easy to use, even for non-standard terminals.

The record retrieval and update menus are also simple. You can print or view records based on a ten-key search. Citation even sorts the records it finds, and shows you the ones with the most matches first. Thus, if you are looking for an article on the Hayes modem, and you can't remember if you gave "Hayes" as a keyword, search for both "modem" and "Hayes." If you did use the keyword, that record comes up first; if not, it is still in the selected group.

During the keyword search, you can view the records on the screen, print them on your printer, and write them to a file simultaneously. You can even select which ones to print or write to a new file from the group of records you are reviewing.

Citation also lets you print out the keyword file, so that you can clean up some of the keywords after you have been using the system for awhile (change "modem" to "modems", etc.). You can print out the whole file, sorted by source, date, and page number.

## Special Features

Most of the above features could be programmed with a reasonably good data base management system, given enough time, but it is unlikely that anyone would want to go to so much trouble for this type of application. The people at Eagle Enterprises added a few extra features which I think are extraordinary, and other software firms should take a lesson from Eagle.

The biggest problem with database applications where the keys are kept separate from the data is what happens to the user when the key file becomes corrupted. Generally, the words "go to your most recent backup" come to mind, but Citation lets you rebuild the keys with little effort.

I am especially attracted to the concept of keeping all the records in one data file. When you search for a set of keywords, you probably don't care whether the records you find are for a book, or an article, or an advertisement. I might hesitate to repeat a search process three times to find all the information on a particular subject, and Citation makes the repetition unnecessary. (Of course, the utility of Citation's storage method depends on the application you are using it for.)

Another wonderful feature of Citation is the manual. It is clear, direct, helpful, creative, and easy to follow. There is a table of contents, index, and many pictures of the screen. The description of each command tells you which files are used by the defined command, what the screen will look like, and how to recover from errors.

The manual constantly reminds you of the importance of backing up your data diskettes, but Eagle doesn't stop there. Each time you exit Citation, it tells you which files you have updated, so that you know exactly what you need (and, more importantly, don't need) to back up. This is a particularly thoughtful feature which can save the user much time.

## Conclusions

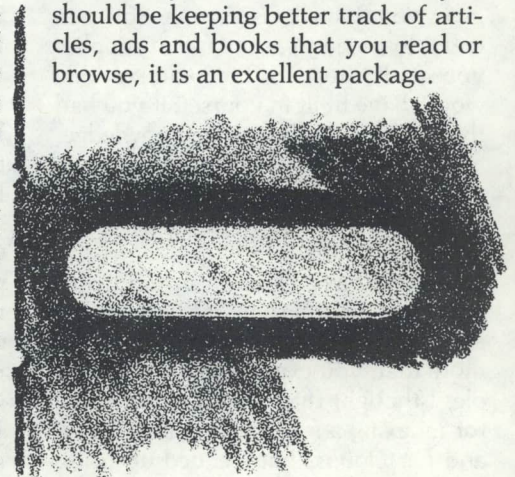
As you might guess, I highly recommend this package to anyone who wants to keep track of everything they read, or to anyone who needs organized references to literature. I can't, however, recommend it without a major reservation: the price.

Without going into a protracted harangue about the price of software these days, I think it is safe to say that \$250 is steep for most users. If the price were \$100, Eagle Enterprises would wear their fingers off copying and shipping diskettes. At \$250, however, many people will have to reject the package, which is unfortunate considering its quality.

However, I doubt if anyone who feels that their time is worth more than \$7.50 an hour could replicate the well-spent effort that went into Citation and come out ahead in dollars. You must decide whether it is worth \$250 for the ability to almost effortlessly catalog all the articles you read, and be able to easily find them later.

Personally, I must keep up with everything happening in the microcomputer field for professional reasons, so I feel that the \$250 is worth it; anyone else who catalogs (or should be cataloging) the information they depend upon will probably agree. If you have doubts, get a demonstration at your local software store.

Citation is one of the best turnkey applications I have seen in a long time. It does everything you expect, it does it with a minimum of effort, and it doesn't let you make fatal errors. If you should be keeping better track of articles, ads and books that you read or browse, it is an excellent package.





Product Information	
Package:	Citation, version 2.0
Price:	\$250.00
Systems Available For:	CP/M systems (8080, 8085, or Z80) with 50K of available memory, dual floppy drives, and a 24x80 terminal with addressable cursor.
Required Supporting Software:	None
Utility Programs Provided:	Crash recovery programs, terminal definition
User Skill Level Required:	Small amount of computer use, although no data base experience is necessary.



# Software Notes

## A Simple Menu Program

Nikki D. Parkyn

Menu driven systems allow an operating environment to become more user friendly. User friendly systems are often essential in commercial environments, where operators should not be troubled by the intricate command strings required by most operating systems.

Certain high level languages support facilities which allow easy implementation of menu structures, like the overlay facilities of Digital Research's PL/I-80. However, in this type of menu implementation, all programs must be written in the same language; therefore, existing utilities and programs for which only the .COM files are available cannot readily be interfaced.

The program I have developed allows .COM files to be executed from a menu. The assembler's location counter allows the tables to be self-maintaining, permitting easier insertion of programs into the menu. The menu program should be implemented so that it is autoloading at warm boot, employing the method that has been described in numerous articles and books. (See *Lifelines/The Software Magazine*, Volume III, No. 1, page 17.) This allows control to be passed to the menu at warmboot and at termination of user program execution. When a valid menu option is selected, the CCP and control is passed to it, invoking execution. The program will currently allow a maximum of nine menu options, but could easily be modified to handle more.

Programs can be inserted by adding them to the menu screen display, execute string table and execute string start address table, as shown in the sample program.

The values of the equates MEMSIZ, SYSBAS, CLSCR and others may need modification to suit your system.

The programs in the sample are essentially for example purposes, since they do not all return via a jump to location 0. Those which do not will obviously not reload the menu program.

```

;*****
;*
;* MENU PROGRAM FOR CP/M
;* N.D. PARKYN
;* 6 PORTLAND RD.,
;* RONDEBOSCH 7700
;* CAPE TOWN,
;* R.S.A.
;* COPYRIGHT (C) N.D.PARKYN
;*****
;-----*
;*
;* NO COMMERCIAL OR MONEY MAKING VENTURES MAY BE
;* ENTERED INTO WITH REGARD TO THIS SOFTWARE (IN
;* ITS PRESENT FORM OR MODIFIED).
;* IT IS INTENDED FOR PERSONAL USE ONLY AND ALL
;* COPIES MUST INCLUDE THIS NOTE. THE AUTHOR'S NAME
;* AND COPYRIGHT AS LISTED ABOVE.THE AUTHOR RETAINS
;* THE RIGHT TO USE THIS PROGRAM FOR ANY PURPOSE
;* HE MAY DESIRE (COMMERCIAL OR OTHERWISE).
;*
;-----*
;
MEMSIZ EQU 56 ;CP/M MEMORY SIZE
SYSBAS EQU 0 ;SYSTEM BASE ADDRESS
CCP22 EQU 3400H ;OFFSET ABOVE CCP BASE, 20K V. 2.2
NULL EQU 00H ;NULL CHARACTER
CLRSCR EQU 1AH ;CLEAR SCRN TERMINAL DEPENDENT
BS EQU 08H ;BACKSPACE CHARACTER
CR EQU 0DH ;CARRIAGE RETURN
LF EQU 0AH ;LINE FEED
FNX1 EQU 01H ;CONSOLE INPUT FUNCTION
FNX9 EQU 09H ;PRINT STRING FUNCTION
FNX14 EQU 0EH ;SELECT DISK FUNCTION
DRIVA EQU 00H ;SELECT DRIVE A
BDOS EQU 05H ;CP/M VECTOR TO BDOS
;
CCPBAS EQU SYSBAS+CCP22 ;BASE OF CP/M 2.2 (20K)
BIAS EQU (MEMSIZ-20)*1024;OFFSET FOR SYSTEM
;
CONBUF EQU CCPBAS+BIAS+07H ;START OF CONSOLE BUFFER
TPABAS EQU SYSBAS+100H ;BASE OF TPA
CMDPTR EQU CCPBAS+BIAS+136 ;COMMAND POINTER
TRFADDR EQU CCPBAS+BIAS ;CCP BASE SYSTEM ADDRESS
;
ORG TPABAS ;PROGRAM LOAD ADDRESS
PAGE 0 ;DISABLE PAGE COUNT (MAC)
START: MVI C, FN14 ;DISK SELECT FUNCTION
MVI E, DRIVA ;SELECT DRIVE A
CALL BDOS ;INVOKE IT
LXI D, MENUSCR ;DISPLAY MENU
MVI C, FN9 ;PRINT STRING FUNCTION
CALL BDOS ;INVOKE IT
MVI C, FN1 ;CONSOLE INPUT FUNCTION
CALL BDOS ;INVOKE IT
CPI '1' ;TEST IF NUMERIC
JC START ;NO RE-DISPLAY MENU
CPI ':' ;
JNC START ;NO RE-DISPLAY MENU
SUI '0' ;CONVERT TO BINARY
CPI MAXOPT+1 ;ALLOWABLE OPTION
JNC START ;NO RE-DISPLAY MENU
DCR A ;FORM TABLE OFFSET
CMP A ;CLEAR CARRY
RAL ;MULTIPLY BY 2
LXI H, PGMTABL ;GET TABLE BAS
MOV E, A ;OFFSET INTO E
MOV D, 00H ;CLEAR D
DAD D ;ADD OFFSET
MOV E, M ;GET LOWBYTE
INX H ;BUMP POINTER
MOV D, M ;GET HIBYTE
XCHG ;TRANSFER TO HL

```

(continued on page 26)

# Feature Access Manager: Searching And Other Necessities

Bruce H. Hunter

Last month we took our first look at Digital's new data storage and retrieval system, Access Manager, concentrating on the data and index file creation routines and updating the files. While the article was being printed I have looked at the rest of AM-80, and found it impressive. The search routines and functions are well thought out, and they function far better than I had anticipated. It seems to have been Digital Research's intention to make Access Manager as powerful as they could, while at the same time as easy to implement as possible. As far as I am concerned, they have succeeded.

Access Manager is intended to handle small to medium-large data bases. In a single user system, it can open twenty data files of eight megabytes each, simultaneously, and at the same time it will hold open ten index files. In an MP/M environment, it can tolerate forty data files of 32 megabytes each with forty open index files. AM-80's ability to access the data base at lightning speed is nothing short of phenomenal and was the subject of the last article.

To demonstrate the flexibility of AM-80, it was my intention to write a "nice little program" for this series, just a little larger than the trivial program of the last article used to bring up the files and add to them. I soon discovered that to even partially demonstrate AM-80's power and flexibility, it was necessary to write over six hundred lines of skeletal code. So this code, and the explanations for it will appear next month.

Before getting into the program, let's take a look at the search routines. Access Manager provides a wealth of search related routines. They can search for key values whether the keys are exact or approximate values and are able to start at the beginning or end of the index file; they can start at either a particular key (target key) or the key presently indexed, and can move either forward or backward through the index file.

## Search Routine Parameters

All search routines require that a number of arguments be passed to them. To be consistent with Digital's manual, I will use their nomenclature to define these arguments:

- KEY** an integer value assigned to the key or index file being referenced
- DLOCK** an integer value indicating the lock or unlock operation requested
- FILE** an integer value assigned to the data file being referenced
- DRN** an integer value indicating the data record number
- IDXVAL** a string value, used as an output parameter to return the key value found by the appropriate search routine (see the note on IDXVAL)
- KEYVAL** a string value representing the key value to be searched by the routine in the index file

Most of the above parameters are either self-descriptive or have been discussed before, but IDXVAL and KEYVAL require a bit more explanation. Keys may be either string (alpha) or integer (numeric) by definition. The parameter KEYTYP defined which was to be used. Its length was defined by KEYLEN. All key values are passed to AM-80 as strings. If numeric (integer, single precision, etc.) values are used internally within the program, they will have to be converted to string values before being passed as parameters to AM-80.

Numeric key values in AM-80 are expected to be passed to the routines that call them as ASCII string representations of Intel hexadecimal code. OK, I didn't get it at first either. The programmer must first convert the number to be input into hexadecimal. Next he must turn the byte order around to put the

least significant byte first, most significant byte last, and next most significant byte next-to-last, and so on. This odd anachronism keeps the bytes in stack order. Now the programmer must convert this into characters, using the order described. For example, 1025 (decimal) becomes 0401H. This is in turn reversed into 0104 and stored as "0104".

Why? At first it seems like an awful lot of work, and reconversion must follow. However, numeric values stored this way require substantially less space than if they had been stored in ASCII. The 1025 example above would be stored as "31303235" in ASCII, taking four bytes. In hex, numbers less than 256 are stored in a single byte. A number like 65535 will take only two bytes, on so on. The larger the key, the greater the storage savings, if the number of records are great enough. You can either enter all your numeric keys as alpha, by setting key type to 0, or be prepared to do the programmatic housekeeping of converting number keys input to Intel hex format. (And, as already noted, keys entered as Intel hex will have to be reconverted on output to human readable ASCII form.)

On the other hand, if the key file is not all that large, you may well ask if this labor is worthwhile. If the numeric key were simply converted to character, by CHR\$, or a similar function, and stored as a ASCII string, it would be stored in ascending "sorted" order as alpha characters are. Additionally, duplicate key values can be assigned (but not to straight numerics). Should a numeric key be passed to the key file as a straightforward number string, the system will store it in ASCII, trusting that you did your homework. However, when you go to retrieve it with, say a FRSKEY followed by a NXTKEY in a loop to return it in order, it all goes down the tubes. The reason is that it has returned what it thought to be hex ascending order, sorted as last byte, first byte.

Numeric key values are tricky, and if you are having trouble understanding them, join the club. I had trouble, too. The concepts are difficult to under-

stand and the manual wasn't particularly clear in explaining numeric key values. I hope this explanation helped, because you must learn these principles in order to use Access Manager effectively.

## Notes On IDXVAL And KEYVAL

---

The parameter KEYVAL passed to most index file routines is simply the key value targeted by the program or program input. In other words, it is either an exact image or a close representation of the key being searched for.

The parameter IDXVAL, on the other hand, is supposed to be the actual value returned by the search of the index file. Your reaction may be, "How is it going to be the value returned by the routine when it is an input argument?"

In fact, the index value is always a string value (as mentioned in the section on key values) with a constant address and length. IDXVAL seems to be a buffer for the index value. In programming languages like BASIC (or CB-80), which do not declare their variables, buffer storage must be "created" by assigning a string of blanks (020h) of sufficient length to hold the actual key value. I presume that in PL/I, declaring IDXVAL to be of fixed length and to be static would accomplish the purpose, but setting an initial value of blanks to the KEYLENGth can't hurt. When IDXVAL is returned, if its KEYLEN is longer than its actual length, it will be padded to the right of the string value with blanks.

## The Search Routines Themselves

---

The routines discussed here are used in the chart of accounts program which will be presented next month. The functions will be covered before getting into the program discussion.

GETKEY is straightforward, going right for the key. In KEYVAL, the target key is passed to the routine along with the key and data file numbers and the lock request. The "Catch 22" of this function is that the match must be exact, and nothing less. In the program listing GETKEY is used to find an exact match of the account number, a prac-

tical use for the function since the account number is exact, and an exact match is not an unreasonable request.

SERKEY is a bit more devious than GETKEY. It looks for the first key (in key sequential order) which is the same as or greater than KEYVAL, the target key value. This function takes all the same parameters as GETKEY with the addition of IDXVAL, the output value found in the buffer. Searching for inexact values such as account names, or any sort of name for that matter, is an ideal application of this function. We can never hope to find a name stored exactly as we would hope to find it. If we are looking for "Ashton Tate" and the index is stored as "Ashton-Tate", GETKEY will return a zero indicating no such key. Since the dash "-" is an ASCII 058h and greater than the blank ASCII 020h, SERKEY would indeed return the name.

I hate to keep dwelling on the idiosyncrasies of IDXVAL (the index value output argument), but it can utterly confuse the program if not understood. We pass IDXVAL to the appropriate routines as a string of blanks of the length expected of the key by KEYLEN. If no equal or greater value can be found, the function will return a zero and IDXVAL will be returned as a string of blanks as it was passed. If the value is found, it is returned in IDXVAL, and if IDXVAL is longer than KEYVAL, the target value, it is padded to the right with blanks. This situation illustrates the need for a software tool to remove the trailing blanks (thank you Kernighan and Plauger).

FRSKEY takes as parameters the data and key file numbers, the lock condition, and our old friend IDXVAL. It returns the data record number of the first value in the index file.

LASKEY takes the same parameters as FRSKEY and returns the last key value in the specified index file. Since the index files are in key sequential order, the utility of the last two functions takes on new importance. Starting at the first or last key and reading the keys to the opposite extremity will give, in essence, an ascending or descending sort, respectively. Slick, really slick.

Now that we have found how to get to an exact value, how to get to an inexact value, and how to get to the first and last keys, all we need to make the func-

tions complete is the ability to go forward or backward through the index. The only problem is whether to move relative to the present position in the index or to the value of the target index. AM-80 doesn't let us down, as the next four routines show us.

NXTKEY returns the data record number of the next key relative to the present file pointer position. It and PRVKEY take as parameters the data and key file numbers, lock request and the output parameter IDXVAL. It returns the data record of the next key, or, rather than dying trying, gives a zero if there is none.

PRVKEY returns the key previous to the present to the index file position.

AFTKEY fills the void, giving us the key following the target key KEYVAL. KEYVAL is the only other parameter needed that NXTKEY did not have, since NXTKEY didn't require a target key.

BEFKEY searches in the other direction looking for the key which is less than the target key KEYVAL, taking the same parameters as AFTKEY.

There are the search functions, in a nutshell. Contemplating the power of these routines to jump about the data base and retrieve virtually anything (as long as it is in an index), the word "awesome" springs to mind. We in the micro world are used to living with restrictions, so when something like this comes along, the feeling of power is great!

## A Brief Aside

---

I was having coffee a few days ago with Bill Hogan, my good friend and micro systems analyst. Bill has a great deal of expertise on data base systems and is a dBASE "guru." We kicked around relational vs. hierarchical data bases for a while and compared dBASE's B+ tree organization and AM-80's, (both hierarchical).

The discussions distilled down to the fact that whether the relational or hierarchical nature of a data base system should be of little importance to the user. The efficiency of the system's housekeeping and the speed with which it searches are the ultimate factors in the worth of a data base system. Here  
(continued next page)

AM-80 has it all. If AM-80 is taking any time to "thrash" it is not apparent on my system. CompuPro's DMA disk controller and a pair of DT-8's don't hurt, but there can be little doubt: AM-80 is efficient.

Just to sidetrack for a bit, we've been talking about B-trees as if they were the only search procedure in the world. Let's look at the options. We know if we have a file in sorted order, a relatively simple search (like a binary one) will recover data in a reasonable amount of time, provided the file is not too large. Sounds good, but there are a few problems. The data file must be in high speed memory, it must be in an array, and of course it must be sorted. There are efficient sorts, including "canned" sorts such as SuperSort.

The problem is that the program user must always sort the data base before searching it. Perhaps the fastest search is the hash search. Hashing provides an algorithm that will produce a semi-unique value for any key hashed. Any subsequent hashing of a target key will locate on or near the similar key. Problems? The area allocated for the hash table must be large enough to provide for thrashing, but not so large that memory is wasted. The table area must be allocated statically, a procedure which lacks flexibility. Theory aside, it appears as if there is no substitute for the speed and efficiency of the B-tree system.

## File Recovery Routines

I was going to examine search routines only, but as the program got longer, it became necessary to check out the file recovery routines. While it is impossible to guarantee that a file will never be compromised (a nice term for blown), the file recovery functions are guaranteed to repair it. If a file has been "corrupted", you are faced with the undeniable fact that the program cannot open the file(s). To get the program back on its feet, you need an error recovery algorithm. The internal function `ERRCOD()` will return an error code of 70. Use this to trigger the functions `OPRDAT` and `OPRIDX` to reconstruct the file(s), and they will be reconstructed in the condition they were in before they were operated upon by the last iteration of the program. Can you imagine having a meg or so of file open without it? Let us look at these

functions a little more closely.

### OPRDAT

A data file loses its integrity by being opened and not being closed by `CLSDAT` or `SAVDAT`, the file closing or file saving functions. To rebuild the data files, `OPRDAT` reconstructs the file directory to agree with its contents. The parameters to be passed to the function are the data file number, lock request, filename, and the length of the record, the same as for `OPNDAT`. The programmer's guide is not clear on what is returned, but if you read between the lines, the function will return the data file number. In actual practice, the function works as fast as the open routine, and to bulletproof your programs, I would recommend that the program jump to a file rebuilding subroutine or procedure on encountering a "corrupted file" error.

### OPRIDX

This routine does for the index file what `OPRDAT` did for the data file: it restores the index file once it has been compromised. It takes the same parameters as its counterpart `OPNIDX`, the keyfile number, the keyfile name, the key length, the key type, and the duplicate key switch. It returns the index file number.

## Deletions

I came to the point where the sample program could update and search every way to Sunday. It seemed like a good time to make it perform deletions as well. This seemingly simple task turned out a bit more difficult than it might have been. Deleting a data record is accomplished by a function that marks the file for deletion and puts the data file number on a stack for use by the next update. Another function simply deletes the index file. This is simple in theory but in fact a bit more difficult. Let's look at the functions.

### RETREC

`RETREC` does an effective deletion of the file record by marking the record's first byte as "FF". It then places the data record number on the top of the stack of deleted records to be used by the next record number request. It takes as parameters the data file number, record lock request and data record number. It is straightforward in a single user environment, since no locks are used. It

must have a 0 or no lock request, but again, in a single user system, why not?

### DELKEY

This function will delete the index key specified by the data record number. The parameters expected by the function are their index file number, data file number, lock request, the key value `KEYVAL` and the data record number. It returns an update code indicating a successful deletion, a not found, a lock conflict, or an indication of conflict between the data record number and the `KEYVAL`.

Deleting a numeric key is simple, whether passed as a string or on reverse order. The numeric value is absolute, in a sense, and a direct match is easily accomplished. On the other hand, a string key, like a name, presents a bit of a problem. Since `KEYVAL` must be an exact match, care must be taken to duplicate the key from the key source, the record field it was taken from. If it is shorter than the key length, it must be properly padded and justified. Key values are padded to the right with blanks (ASCII 020h). If duplicate keys were used, then the value is truncated by two bytes.

## Statistical Functions

After adding to and deleting from the data base, it would be nice to find out just where we are. A number of functions are available in AM-80 to allow the return of statistical data. The statistics functions are more than a nicety, they keep track of the size of the data base and can easily be modified to yield remaining disk space. The functions that deal with the index files take only the index file number as a parameter.

`NOKEYS` returns the number of keys that in are in the key file.

`NMNODS` returns the number of index file records or nodes.

Going back to the discussion on nodes last month, remember that each record or node in the index file points to a group of data file records. The length of the index files was established back when the files were set up under `SETUP` as four 128 byte sectors (or 512 bytes). The length of the individual keys was

set by KEYLEN. Each key as filed takes its key length plus four bytes. The number of keys in a node can be calculated as the byte length of the index file record length less 10, all divided by the key length plus four or

$$\text{integer}(((\text{number} - \text{of} - \text{sectors} * 128) - 10) / (\text{key} - \text{length} + 4)).$$

(Should appear on one line) In the case of the numeric keys used in the demonstration program to be listed next month, the number of nodes per record is

$$(512 - 10) / (4 + 4) = 62.$$

You can see the source of the B+ tree's efficiency when you recognize that 62 keys are accessed at each node, as opposed to two in an ordinary binary tree search.

### GETDFS

For statistics on data files, GETDFS takes the file number as the passed parameter and returns the actual size of the data file in records, including the 128 byte header record used for housekeeping by the system. Don't forget to subtract the header record from the total to find the actual number of records used for information. If records have been deleted from the data file, GETDFS will include the deleted records in the record count. To avoid this frustration an alternate function called GETDFU can be used.

### GETDFU

GETDFU returns the number of files in actual use, plus, of course, the ever present header record. Now that we have discussed some of the functions, let's look at the file internals.

## File Internals

For the last few years (on my own systems) I've made a habit of doing a hex dump on new file systems, just to "see what was inside." AM-80 was no exception. It quickly succumbed and dumped chartdb.dat. The dump was not entirely a surprise. The data showed up pretty much as expected in ASCII representation, preceded by a byte or two of intelligence, presumably marker bytes, as in PL/I direct (random) files or Apparat's fixed marked files.

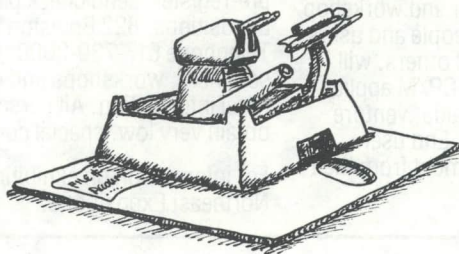
Dumping the key files was quite an-

other story. The dumps were longer than the data file and well padded with 0h's, presumably to make room for additional housekeeping. The numeric and alpha dumps were similar but not identical for the first 200 hex bytes. Apparently AM-80 uses this area for its internal housekeeping. Below this area came a marker byte to keep track of the number of records, a handful of hex zeros and the data record number followed by the key in ASCII. The keys were stored in ASCII ascending order and the data record numbers in "as entered" order. A pair of hex zeros lay between each data record number and its key, which must allow the high order two bytes used by SETDAT and DATVAL to keep track of record numbers over 65,535.

Numeric keys stored as numeric keys were in most/least significant order as they had been passed by the function created by the programmer for that purpose. AM-80 had stored the keys in sort order by entering them sorted according to most significant byte/least significant byte. Like alpha keys, they were preceded by the data record number.

## Parting Comments

Next month we'll put these functions to work in the chart of accounts program I have written for demonstration purposes. ■



DATABASE

©Reli182

Make your systems sales soar!

## STOK PILOT™

A language for creators.

STOK PILOT can make your final product more saleable and keep it sold.

This unique language allows straightforward creation of a friendly interface between novice users and CP/M for any program application. Using STOK PILOT, a software designer can easily produce a hand-holding menu-driven front-end supervisor with built in tutorials to help guide the novice end-user through a complex application. The actual end-user does not have to know STOK PILOT.

STOK PILOT opens the door for companies to evolve structured approaches to meeting their own training needs.

STOK PILOT can simulate any application program for the purpose of tutoring the user. This allows training and guidance without the necessity of working on and possibly damaging live data. Then, when the user is ready, STOK PILOT can call the actual application program.

STOK PILOT is a superset of the PILOT CAI language. STOK PILOT was designed with the syntax and structure of PILOT because PILOT is an easy to use and easy to learn language.

STOK PILOT can be thought of as an interactive SUBMIT facility. One program written in STOK PILOT can control an entire session without ever entering cumbersome CP/M commands. The commands can be dynamically formed at run time by STOK PILOT.

Menus can be created that will allow the user to select various programs. STOK PILOT can chain to any program or utility written in any language, leaving the entire TPA available, and regain control when the other program ends.

STOK PILOT can check for the existence of a file before it actually calls the application that needs it. Let STOK PILOT see to it that the proper disks have been inserted by the user.

The package includes a well written and indexed 75 page manual. All the instructions are explained in full detail. Many useful programming examples are also included.

Complete 8 inch CP/M format disk and manual retails for \$129.95. The manual is available alone for \$14.95 and is deductible from a future order. NY residents please add sales tax.

Toll free order line: (800) 431-1953 ext 183  
In NY (800) 942-1935 ext 183

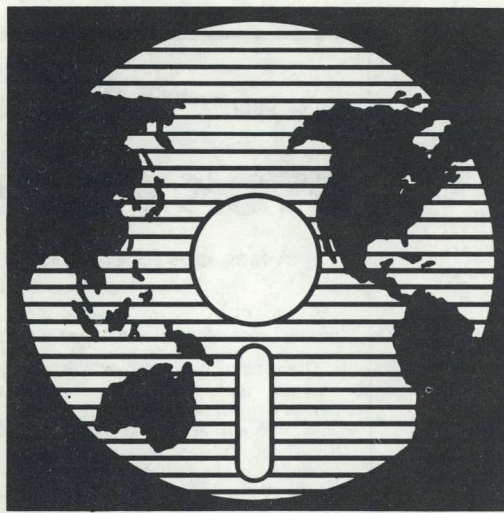
C.O.D. - Mastercard - Visa



Stok Software Inc.

17 West 17th Street  
New York, NY 10011  
(212) 243-1444

Dealer inquiries invited  
CP/M is TM of Digital Research



# THOUSANDS OF CP/M SOFTWARE PACKAGES ARE GATHERING IN SAN FRANCISCO

## CP/M '83<sup>®</sup>

Moscone Center, San Francisco  
Friday-Sunday, January 21-23, 1983

Sponsored by

 **DIGITAL RESEARCH™**

The Creators of CP/M

CP/M '83 is an International Exposition and Conference for the CP/M industry and CP/M users. The exposition portion of the event will be the largest presentation of CP/M based hardware and software ever assembled. Nearly three hundred companies using over six hundred exhibit displays, will showcase the full spectrum of application packages, development aids, peripherals, accessories, publications and services for CP/M based computers . . . making CP/M '83 the largest first year computer event ever.

CP/M '83's Conference and Seminar program will include noted leaders from the industry including Gary Kildall, President, Digital Research Inc.; Sol Libes, Editor, Micro Systems Magazine; Christopher Morgan, Editor-in-Chief, Byte and Popular Computing Magazines; Adam Osborne, President, Osborne Computer Corporation; Tony Gold, Founder CP/M Users Group and Lifeboat Associates; Ben Rosen, President, Rosen Research; Portia Isaacson, President, Future Computing; Maggie Cannon, Editor in Chief, InfoWorld; David Crockett, Senior Vice-President, Dataquest; and Gordon Eubanks, Vice-President Language Div., Digital Research. CP/M '83's conference, seminar and workshop program is designed for computer tradespeople and users. The individuals listed above, plus dozens of others, will conduct informative discussions exploring CP/M applications, technical information, development aids, venture capital programs and software distribution. End user workshops will show users how to get the most from their CP/M computer.

CP/M '83 is sponsored by Digital Research Inc., the Creators of CP/M. Over 650 different companies support CP/M and more than 5,000 companies produce CP/M application packages . . . All of these firms will be represented at CP/M '83 . . . so attendees will see and try out application packages for every profession, plus state-of-the-art programs for word processing, telecommunications, graphics and data base management. Also on display or as workshop subjects are development aids to help you program faster, plus hundreds of CP/M compatible products appropriate for your applications.

At CP/M '83 you can explore the entire world of CP/M under one roof and learn more in three days than you could in six months, any other way . . . and everything which is on display is for sale.

Registration is \$10 for one-day exhibits-only ticket, or \$20 for a three-day ticket which includes admission to workshops, seminars and conferences. Attendees can pre-register thru the mail or purchase tickets at the door. To pre-register, send check payable to CP/M '83 to Northeast Expositions, 822 Boylston Street, Chestnut Hill, MA 02167, Telephone 617-739-2000. Call or write for the schedule of seminars, workshops and conferences plus San Francisco hotel information. All persons attending CP/M '83 can obtain very low, special convention hotel rates.

For information on exhibiting or attending call or write Northeast Expositions.

Name \_\_\_\_\_  
Company (if any) \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_  
Telephone (Area Code) \_\_\_\_\_

Check applicable box—Make checks payable to CP/M '83. Payment must accompany ticket orders.

- Send me \_\_\_\_\_ one-day exhibits-only tickets @ \$10 each per day.  
quantity
- Send me \_\_\_\_\_ three-day exhibits and conference tickets @ \$20 each.  
quantity
- Send me seminar, conference & workshop schedule along with San Francisco hotel reservation information.
- My company may be interested in exhibiting, send me information.

# A Review of SUPERFILE

Bob Kowitt

So you say, "Another database management program? Ho-hum!" Well, not quite. SUPERFILE is a data base management program into which you do not enter data. O.K., then how does data get there, and why?

The usual concept of the data base management systems (as currently available for microcomputers) involves setting up a group of fields which are defined according to the particular type of data we wish to store and manipulate: names, addresses, numbers, phone numbers, inventory numbers, etc. These must be pre-defined and pre-located within each record. SUPERFILE is different. Rather than letting you enter information into preselected fields for later calling up and manipulation, SUPERFILE takes existing text files and makes an index of selected keys. Then, when these keys are entered through the keyboard, SUPERFILE will tell you, by the name you previously assigned to the disk, which disk to put into your disk drive. SUPERFILE then will call up the original document and will display, on screen, printer, or write to your disk, all or selected portions of the text. For writers or executives who do much correspondence, it seems to be a very good idea. And when this information can be anywhere within 20 megabytes of hard disk storage, it seems to be a necessity.

Before getting SUPERFILE, I wrote a file search program for a commercial photographer. With thousands of photos and thousands of subjects in thousands of different situations, finding a particular photo for an advertising client can be a monumental job. I limited my search to four keys and had the entry record limited to 80 characters free style. The program parsed each record entry for the keys logically ANDing the search parameters. It works but is limited, takes more time, and requires fixed 80 byte records even if the entry only needs 10. SUPERFILE, in practice, removes these limitations, permitting elaborate descriptions of the photographs where necessary and brevity where it suffices.

Upon opening the package you get from FYI Inc, of Austin, Texas, you will be pleased to discover a manual that was put together by someone who appears to have struggled with understanding software documentation. The manual is printed in rich blue on white stock and is complete with table of contents, index and large plastic-coated tabs for ease in locating the area you want. Sorry, Zoso, it is a loose-leaved "wedgie" but since it is pretty full, it is relatively straight. I did not count the pages but the contents of the manual are about two centimeters thick - a good deal of paper, well used.

Under a 30 point heading on the instruction pages are three main layouts used throughout the book:

- 1) the left page is headed COMPUTER SAYS, the left side of the right page says YOU ENTER and the right side of that page is COMMENTS on the particular item being explained. Between each item is a thick blue line to keep your attention within the proper area,

- 2) the left side shows the MENU as displayed and the right page explains in detail the options offered,
- 3) explanatory text in standard page layout.

All this gives one a secure feeling when reading the manual for the first time. The instructions are easy to read, not full of cutesy "Whoopee" and sophomore game-talk, and also not technical to the point of head-spinning. There is a lot of information here and it should be read. However, you don't have to read it all at once to see how SUPERFILE works. There is a tutorial section to get you up and running with the demonstration files.

Setup is minimal. SUPERFILE does not use direct cursor addressing or enhanced video modes, if these are available on your terminal. The installation procedure will install the clear screen codes for seven terminals by menu. If yours is not included, use the '0' menu option and you will be able to install your own clear screen code in decimal numbers.

If you are adding SUPERFILE to your library, you probably have at least one disk of letters that you have written with some word processor. In my case the word processor is WordStar. To make a data base of these existing letters you will have to load each of them and make some modifications. The beginning of each document must be marked with '\*C' and the end with '\*E'. Of course, if you modify your original text in this manner, some means must be provided to prevent these marks from printing. WordStar provides this ability with the "double-dot" comment line. In WordStar, enter the beginning and end markers with ..\*C and ..\*E and you're home free. There are two more markers available, one required and the other optional. The user is allowed to select as many keys as desired for each text entry to the data base.

SUPERFILE will find the entry using these keys. At the end of the file, but before the final \*E, start a line with \*K. This indicates the beginning of your key list. Each key must be separated with a "/" and there may be more than one line of keys. This key may consist of one or more words, but a warning is in order. If you search using a multiple word key, the key must be exact or SUPERFILE will not find it. The number of keywords allowed for each data base is limited to 250 words per entry and 3000 words per data base with 64K RAM memory. You can also have as many whole numbers (from 0 to 31999) as you wish used as keys. Therefore, some discretion should be used in the selection of keys. For this reason, the authors suggest you keep a hardcopy list of keys available when preparing the text, to avoid plurals and similar keys. For example, LIFELINE and LIFELINES, DATABASE and DATA BASE, BALL POINT PEN and BALLPOINT PEN. To avoid different types of date entries and bypass the word-key restriction, use a different date, numbers in all cases.

Modify all texts in each category before creating your data base, since batch processing is possible - and easier than going at it one at a time. At this point one of several decisions must be made. If you are certain that there will be no more

(continued next page)

than 99 separate texts entered per data base, you can proceed with creating and adding to the data base. However, while this restriction may be no problem for large texts such as magazine articles, it probably will not be practical for data on hard disks or short text files. For these situations, another technique is required.

When SUPERFILE creates a data base, it recognizes everything between \*C and \*E as a separate entry. Therefore, within one text file, as we recognize it, we can have many SUPERFILE entries. Because of this, when more than 99 are anticipated, the files should be concatenated before indexing. The simplest means of doing this would be:

PIP newfile.txt=file1, file2, file3, file4.....

If you run out of command line room, stop, let PIP run out and then:

PIP newfile.txt=newfile.txt, file9, file10, file11, ....

If you are starting a new file, or adding to a text file such as a personnel file or an inventory description file, use your text editor and enter all the data, using \*C, \*K, and \*E for each entry description. Remember, changing a text will require completely reindexing the data base, so you should probably do this when you have several changes to make and not just one.

When your text is prepared, invoke SUPERFILE with SF and follow the menus. You must decide on the data base file name, the data base title (max. 64 characters), the date (for an updating record), and up to 64 characters of commentary. If you are using a floppy disk system, there may be many different disks of original text material. Each disk must be given a name. When retrieving data, SUPERFILE will tell you, by name, which disk you must insert into the disk drive to examine the text file itself. After you enter the Raw Data filename or filenames (you can use CP/M's \* wildcard specification), SUPERFILE will create the data base and display the number of keywords and entries added. The raw data filetype will have been changed to Dnn where the nn is a number from 00 to 99. For this reason, you will need file concatenation when working with floppy disks and you'll need it even on a hard-disk. You cannot afford to use 2K of disk space (or 16K on a hard disk) to store 20 lines for each employee in a personnel file.

Before searching for an entry, check over the list of keys by displaying the dictionary. The keys used must match those in the index file exactly, so they must be correct. This choice allows many options:

- 1) Full dictionary
- 2) Select Range
- 3) Partial Match

As usual, the manual, with its two page format, has full information about each option.

*Option 1* is self explanatory.

*Option 2* allows entry of a low and high range of words for those times when you don't know the exact key word but can get close.

*Option 3* will allow reading of keys that are "something

like" what you have in mind. With this one, you can search for keys than contain "JO" and find JOB, JONES, BANJO, and PANMUNJOM.

After choosing the type of search, the output options are:

- 1) Screen
- 2) Printer
- 3) Disk File

Should you name a disk file that already exists, there is a warning before you overwrite the existing file. The authors have been very considerate of us humans in helping us avoid catastrophe.

Now that the available keys are known, it is time for a real search. The keys are entered one at a time in response to prompts from the system. Entering a key which is not in the dictionary causes a "Not recognized" message with a list of those keys that are close in the list. The word "close" takes some understanding. This list is not necessarily those keys that sound like or look like the incorrect one. They are close on the list in the dictionary. SUPERFILE could use some form of "SOUNDEX" program to find those entries that "sound like.....". The keys, when accepted as being in the dictionary, will be linked with one or more logical links, AND, OR, /NOT, supplied by the prompts. You are allowed up to 64 keys with "AND", 32 to 64 when using "ANDandOR" and the use of "/NOT" occupies the space of one key. This seems sufficient for the most precise search. If the search string will be used more than once, you can save it with an appropriate comment applied for future clarity. You can then invoke the same search by entering /S. You will be prompted for the file name of the search string. A hard copy list of search strings will be necessary for this use.

Once the data base is searched, SUPERFILE displays an output menu and the number of entries found that fit the required parameters. Again, as is often the case during operation of SUPERFILE, several options are available. Either the entire file or portions may be displayed. During the creation of the text, another "\*" may be inserted anywhere in the text. One of the options is display of the text between "\*C" and "\*". If you have placed the "\*" just after the address header of a letter, for example, SUPERFILE will display all the headers of all the letters fitting the selection criteria. One of the options is the display of only the entry headers, that is, disk names and filenames fitting the criteria. At this time, the search string used can be saved for future searches with the /S command.

The manual is generous with practical uses of SUPERFILE. They illustrate:

- Records of Production Change Orders.
- Vendor or Client/Customer Records
- Contract Records
- Tickler (reminder) File

\*\*\*\*\*

Once your data base is established, there are some additional utilities supplied with SUPERFILE. They are Sort/Merge, File Split, File Rename and Character Change.



If the file consists of a concatenation of many individual entries, it could be useful to have the entries with each file sorted alphabetically, by number, by date, by zip code or by any other item you have placed at the top of the entry. Since sorting is based on the first part of the text entry, planning the text properly can allow SUPERFILE to sort by the parameter of your choice. Remember, if you enter the sort parameter at the top of the entry in the following way, WordStar will ignore it:

```
..11554
..*C
```

The sort is a character sort. Sort parameters must be the same length and have the same format: 001,002...075...999. The sort parameter may be up to 65 characters long; but while this will be the most accurate sort, it will take more time and use considerably more memory. Lower case can be transformed to upper case for the sort, at the option of the operator.

The Merge facility, like the other components, contains options. After a sort/merge of files, the output can be to one large file or split to multiple files. If you should want to sort/merge customer files, hardware suppliers and service organizations by zip code and then split them into several zip code areas, SUPERFILE will do it with the Sort/Merge facility.

The File Rename option is not just CP/M's REN. When text files have been indexed, the filetype is renamed Dnn. If you want to add to the index, just do so with the new file names. However, before the entries themselves are altered by changing keys or concatenation with new data, the filetype should be renamed. File Rename does this individually or by renaming all the Dnn files to TXT. After alteration or concatenation, reindexing replaces the Dnn filetype.

The Character Change option will allow up to 255 character changes in a single pass. This is simpler than using your text editor or word processor global change mechanism to make one change at a time. In addition to alphanumeric changes (a to A, C to V, etc.), non-printing characters can be also be changed so that the control characters of one word processor can be modified to those of another. These are entered as decimal values.

In addition to the usefulness of the software they are offering, the FYI folks in Austin, Texas also have joined the growing list of enlightened and user-aware vendors who are offering a money-back guarantee. SUPERFILE is available with a "no risk" 30 day guarantee to allow actual testing under your own conditions with your own text files. When you get your software from them, you will be in for a surprise. The package does not include a limited-use demo version for your trial, but an actual, complete working copy. Obviously, these vendors not only have confidence in their package but in the basic honesty of their customers. There is some applause due them for that. I hope other software vendors take notice.

For people who use words, SUPERFILE can be a valued addition to their software library of tools. Its free-form data storage is easy to use and the output options are numerous enough to adapt to the user's requirements with very little computerese necessary.

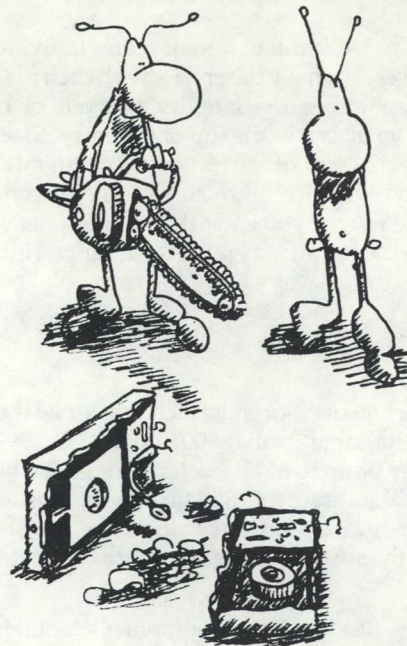
**TABLE I  
Facts & Figures**

<b>Package and Version:</b> SUPERFILE Version 1.13
<b>Available from:</b> FYI, Inc. 4202 Spicewood Springs Road #204 (doesn't that sound great!!) Austin, Texas, 787595
<b>Price:</b> \$195
<b>Operating System Requirements:</b> Z80 microprocessor with CP/M-80 or MP/M-80
<b>Memory Requirements:</b> At least 48K RAM
<b>Other Requirements:</b> At least two 8" disks or hard disk with up to 16 logical drives.
<b>Available Formats:</b> 8" IBM Single Density IBM-PC with Baby Blue card
<b>Auxiliary Programs Required:</b> A text processor or word processor for preparation of ASCII input text.
<b>Utility Programs provided:</b> A sort/merge module described above. A file split module to allow splitting files into two or more categories. File rename module and a global character change module.
<b>Portability:</b> Portability between systems with similar disk capacity should prove no problem if logical disk drive names are the same.
<b>User skill level required:</b> There should be no difficulty in setting up the system, even for a novice. However, a great deal of thought should be given to the manner in which keys and file names are assigned.
<b>Disk capacity:</b> If used only for key dictionary and index: approx. 7000 entries on 8" SSSD disk approx. 2500 5" SSSD disk 20000 on hard disk is claimed but the size of the hard disk is not specified by FYI.
<b>Search Speed:</b> About 100 items per second assuming Z80 @ 4mhz., 10 keys per item.
<b>Source language:</b> CONVERS, a proprietary language of FYI.
<b>User Support:</b> An "800" assistance number
<b>Trial Policy:</b> Full scale program on money-back guarantee.

(continued next page)

**TABLE II**  
**Qualitative Factors**

	Rating *
<b>Documentation:</b> organized for learning organization for reference readability includes all needed information	6 4 6 5
<b>Ease of use:</b> initial start up setting up text files application implementation	6 6 6
<b>Error recovery:</b> from input error from data media damage	6 2
<b>Support:</b> for system implementation	6
* Ratings in this table will be in a 1-7 scale where: 1 = clearly unacceptable for normal use 4 = good enough to serve for most purposes 7 = excellent, powerful, or very easy depending on the category	



© Ritz

I JUST INVENTED THE HALFHEIGHT FLOPPY DRIVE...

## BASIC/Z...

the ultimate CP/M compiler!

- Generates native code (8080/Z-80) for fast execution - 16 bit versions soon
- Sort verb is unmatched by stand-alones. 2000 elements in two seconds!
- Alpha-numeric labels, variable and function names of any length
- Chain program segments which share variables declared common
- Five data types - binary/BCD/string
- BCD floating point math - never a "round-off" error - precision is program definable from 6-18 digits
- Full function program editor tests syntax as you type
- Recursive, multi-line, multi-argument user defined functions
- **No royalties - No run-time charges**
- Dimension arrays dynamically (to an expression) and selectively erase
- Screen oriented editing of console input at run-time (cursor left/right/start/end, delete left/right/line, insert/change mode, and input masking available)
- Push/pop subroutine stack
- Trace and single-step debugging
- Multi-tiered error trapping even handles BDOS errors
- Cursor addressing, reverse and blinking video, erase and more are supported from source code level, with virtual hardware independence
- An extended library of over 200 "key-word" functions

For free brochure  
and mini-manual:

**System/z, inc.**

P.O. Box 11  
Richton Park, IL 60471  
(312) 481-8085

\* a trademark of Digital Research

# System/z, inc.

# MICROCACHE

**MICROCACHE is not a RAM DISK.**

**MICROCACHE is a totally new concept which, whether you use or deal in or manufacture microcomputers, can increase your file accessing speeds by 50,000%! It helps you out-perform your rivals and puts you so far ahead, you're practically printing money!**

**MICROCACHE does it by monitoring and automatically storing 'most used' disk records in a separate memory area for ultra high speed RAM-to-RAM transfer. (We also invented SILICON DISK - the world's first RAM DISK for microcomputers - to provide an additional ultra fast disk drive which is capable of being accessed 1,000 times faster than ordinary disks).**

**MICROCACHE dovetails into most existing systems without difficulty. It simply loads itself into your systems and runs in the background speeding up disk accessing for all applications. It is easy to use, virtually transparent and, by reducing disk activity, dramatically improves the performance, reliability and cost effectiveness of your microcomputers.**

**If you want to speed up your microcomputer and your earnings, talk to us.**

## Microcache Technical Description

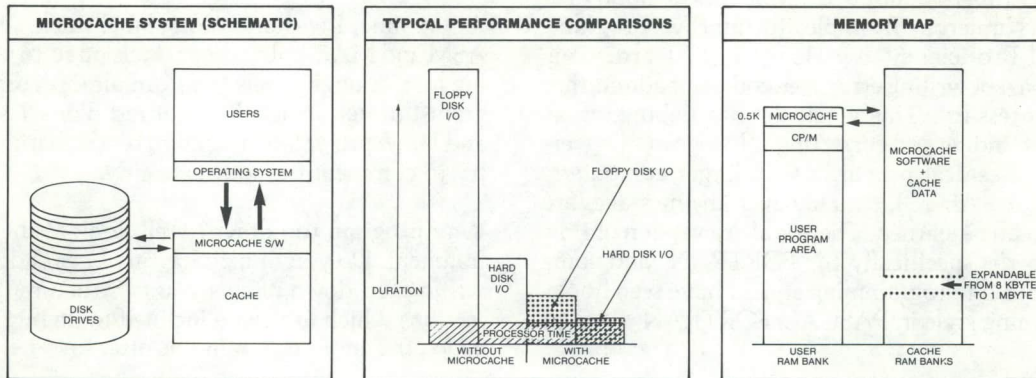
Microcache will dramatically improve the performance of your microcomputer by reducing the number of disk accesses to the bare minimum. Disk accessing is usually the prime cause of slow response, even with hard disks. Surprisingly, the vast majority of these accesses are not in fact necessary. The overall speed increase achieved by such a technique depends on the application and the hardware, but is typically in the range of 2 to 50 times as fast and can be as much as 500 times as fast for some applications! Disk reliability, a major problem on computers, is also greatly improved.

The Microcache software prevents unnecessary disk accesses by means of a highly intelligent and automatic buffer (cache) situated between the disk drives and the user. Sophisticated algorithms are used to ensure that those disk records required most often by an application are automatically stored in this RAM 'cache'. Consequently, on subsequent disk requests they are transferred at ultra-high speed (RAM to RAM). The software monitors use of the disks and 'learns' what to hold in the cache. In

addition to this caching, a track buffering system is included to increase the speed of the few disk accesses that are actually required.

You can use as small or as large a cache as you wish (up to a maximum of 1 megabyte). The more RAM added, the greater the speed improvement, but even small amounts have a very significant effect. An extra 48Kbytes is usually adequate for single user systems. Computers with more than 64K of RAM as standard (e.g. most 16-bit machines) may not need any additional RAM at all. The Microcache software runs with the existing operating system and is transparent to the user and his programs. The only effect is a very marked improvement in speed. Facilities are included to enable selective locking/unlocking of disks, files and directories if required.

Microcache is available for computers using the CP/M, MP/M, CP/M-86, MP/M-86 or MSDOS operating systems. The machine must, of course, be capable of accommodating additional RAM.



## Silicon Disk Technical Description

The Silicon Disk is a disk emulator that configures additional RAM (up to 8 megabytes) to appear to the user to be an extra ultra-fast disk-drive (up to 1000 times as fast as a floppy disk!). It is accessed by the user and his programs like any other disk drive.

Facilities include memory diagnostics, autostart routines and the ability to alter the mapping of logical to physical disk drives.

The Silicon Disk software will operate on most CP/M computers capable of accommodating additional RAM.

**We, at Microcosm Research, are imaginative, competitive and cost conscious. We make products to help you become more imaginative, competitive and cost conscious.**

## MICROCOSM RESEARCH LIMITED

26 Danbury Street, London N1 8JU, England.

Tel: (01) 226 9092

Telex: 24263 TARDIS G.

Now also available  
for 16 bit  
systems

# 8080 Assembler Programming Tutorial, Debugging

Ward Christensen

Bill Precht, a friend who works as a computer consultant, once walked into a client's office and asked what the client was doing. The reply: "Bugging. Tomorrow I'll be debugging, so today I must be bugging."

So it goes. Unless you have a photographic memory, flawless logic, and the persistence of a hungry mosquito, you will eventually write an assembly language program that does not work. Or if you are like me, you will always write programs that don't work, at least the first time, and perhaps for some considerable time. My favorite acronym for any "finished" program I have is OLB - One Last Bug.

## Do It Right In The First Place

One way to minimize debugging is to spend more time up front, i.e. in the definition of the task you want the program to perform and in the detailed layout of the steps necessary to accomplish that task.

An enjoyable book on the subject of writing good programs is *Programming Proverbs*, subtitled "Principles of Good Programming with Numerous Examples to Improve Programming Style and Proficiency" by Henry F. Ledgard. It is dedicated to the art of writing error-free code, but admits that debugging is necessary. This work defines debugging as "...the process of finding and correcting... [programming] errors". The examples deal primarily with larger computers, and are in PL/I and ALGOL 60. However, the messages are clear, and much can be learned. There is also a version of *Programming Proverbs* specifically for FORTRAN, and some more recent books on programming style. I have seen books about programming style in PASCAL, FORTRAN, BASIC, and COBOL.

## Structured Programming

Structured, or "top-down" design and programming have influenced programming style for more than a decade. In simplistic terms, top-down design means starting with a "big picture" definition of the task, then breaking it down into smaller and smaller pieces, until you are at a sufficient level of detail to do the actual programming. By breaking your problem down this finely, you make each phase (presumably) easier to understand, and therefore easier to program.

Top-down programming refers to programming in a similar style. I find it very useful, because it supports my preference for programming *without* much design.

For example, many years ago I needed a sort program to go

along with the mailing list for the computer club I was treasurer of. I thought of sitting down and writing unstructured line after line of code, until what I had would do the job. Instead, I decided to write the program top-down, designing as I programmed.

I thought about the steps involved - to perform some initialization, read the table of data (zip codes, usually), sort them and write out the table of record numbers that defined the file in that particular order. This easily became the plan for the major opening lines of the program. Following the CP/M-80 obligatory ORG 100H, I wrote the "top level" of the program:

```
LXI    SP,STACK
CALL   INIT
CALL   READ$TABLE
CALL   SORT$TABLE
CALL   WRITE$SORTED$FILE
CALL   MSGEXIT
DB     '++END OF SORT, '
NREC   DB     'RECORDS SORTED$'
```

At the time, I was on a "long label" kick. Digital Research's ASM and MAC both accept labels up to 16 bytes long. So using long enough labels, you can almost comment your program through its labels. That fad didn't last long however, and I now prefer labels of up to seven characters, and usually try to comment the code explicitly.

Why program top down? Well, look at the above program fragment. Do you think it has any bugs? I don't think so. So, writing top down allows you to structure the program into sections which are more inclined to be bug free than if you coded the entire program line after line, i.e. unstructured.

Take a look at the INIT routine:

```
INIT   CALL   GET$FLDNO
        CALL   SAVE$FCB
        CALL   OPEN$INPUT
        RET
```

Again, most likely there are no bugs in this routine. There *could be*, if the call to SAVE\$FCB had been omitted, and therefore the OPEN\$INPUT routine would have wiped out the information which was in the second FCB at 6CH. Structured programming is not perfect, but it *does* help.

## Debugging Techniques

What are the most common debugging techniques?

- **Programmed-in:** You might assemble in some routines that print the contents of various data fields, counters, etc.
- **Debugging tools:** These include DDT as supplied with CP/M-80, or Digital Research's symbolic debugger SID, or others like ZSID, RAID, or DDS (the latter being an interesting full screen debugger from PRS, the Program of the Month Corporation).

As an example of the first technique, consider the way I examined some problems I was having with my disk I/O. I wondered what calls CP/M-80 was making to such routines as home, drive select, sector select, read, write, etc. To find out, I programmed in a simple routine to print H when the home routine was called, Dn when a drive was selected, Snn when a sector was selected, Tnn for a track, and finally R for read or W for write. This information was sent to my console, and looked something like:

```
D1 H T02 S01 R S07 R...
```

Interpreting this, I could see that drive 1 was selected, followed by a home, select of track 02, sector 01, then a read, etc. I was able to solve my elusive problem. DDT or SID cannot be used to unravel a problem like this, because neither has the ability to easily trace such events without burdening me with significantly more data than I wanted. For example, SID "pass points" (the addresses at which the contents of the registers are dumped) with execution automatically resuming would have yielded several pages, rather than several lines, of information.

Another technique I have used is to print each label in a program, as it is executed. For example, I start out my program with an equate as to whether or not I want trace active:

```
trace equ true
```

Then, for each label, I code something like:

```
CHARLP:
    if trace
    call trsub
    db 'charlp',0
    endif
```

I'll make a complete program of that, showing the "trsub" trace subroutine, and show you an example of it running:

```
false equ 0
true equ not false
trace equ true
org 100h
lxi h,0 ;hl=0
dad sp ;hl = CP/M stack
shld stack ;save it
lxi sp,stack;init my own stack
mvi b,5 ;set char loop count

CHARLP:
    if trace
    call trsub
    db 'charlp',0
    endif
    dcr b ;decrement count
    jnz charlp ;loop if more
```

```
fini: ;end of program
    if trace
    call trsub 'fini',0
    endif
;
    lhld stack ;get CP/M's stack
    sphl ;restore it
    ret ;return
;
; trace routine prints label following
; call trsub.
; the label is binary 0 terminated.
; All registers are saved.
;
trsub if trace
    xthl ;get addr, save HL
    push psw ;save all
    push b ; regs
    push d ;
    mov a,m ;get a char
    call type ;type it
    inx h ;point to next
    mov a,m ;get char
    ora a ;is it 0?
    jnz trpr ; no, loop
    inx h ;skip final 0
    mvi a,' ' ;type a space
    call type ; after the label
    pop d ;restore
    pop b ; the
    pop psw ; regs
    xthl ;get hl,
    ;restore ret addr
    ret
endif
;
; routine to type char in A.
;
type push b ;save regs
    push d
    push h
;
    mov e,a ;char to e
    mvi c,wrcn ;type it
    call bdos ; via bdos
    pop h ;restore regs
    pop d
    pop b
    ret
;
stack ds 100h ;stack space
;
wrcn equ 2 ;write console char
bdos equ 5
end
```

Here is a sample execution of this test program, after ASMIing it and LOADIing it:

```
A>test
charlp charlp charlp charlp charlp fini
A>
```

(continued next page)

As you can see, charlp was executed five times, fini once. This technique is nice because it doesn't require any special tools, you can print any message you like, and it can easily be disabled by assembling the program with trace equated to false.

The program should be pretty easy to follow, with the possible exception of my use of XTHL - it isn't one of the more common 8080 operation codes.

XTHL swaps the 16 bits in HL with the 16 bits at the top of the stack. This is ideal for a routine like the trace routine. The XTHL saves HL on the top of the stack, at the same time as it loads HL from the old top of the stack - the address of the label message to print. The print routine then increments HL, finally leaving it pointing past the 00 which indicates the end of the label. This is where I want to return to, so another XTHL restores my original HL, and puts the return address back on the stack.

## Debugging Tools

An alternative to such "planned" debugging is the use of the CP/M-80 tools like DDT. Let's take another look at that same program, this time with DDT tracing its execution.

When using DDT, you do not have a way to readily determine where the various routines are. You will either have to recognize them from their approximate location in memory and their context, or by looking at a listing file of the assembly of your program.

Here is the DDT session tracing TEST.COM. I cut the long trace lines in two, splitting them between "D=nnnn" and "H=nnnn" so they would better fit the columns of this tutorial. The trace is pretty well self-explanatory, showing you the contents of the PSW flags (C=carry, Z=zero, M=Minus, E=even parity, and I=interdigit carry for decimal arithmetic), the Accumulator, BC, DE, and HL registers, the stack pointer, the program counter, and finally the instruction at that address.

```
A>ddt test.com
```

```
DDT VERS 2.2
NEXT PC
0180 0100
-a1000
1000 lxi sp,1100
1003 call 100
1006 rst 7
1007
-xp
P=0100 1000
-tffff
COZOMOE0I0 A=00 B=0000 D=0000
H=0000 S=0100 P=1000 LXI SP,1100
COZOMOE0I0 A=00 B=0000 D=0000
H=0000 S=1100 P=1003 CALL 0100
COZOMOE0I0 A=00 B=0000 D=0000
H=0000 S=10FE P=0100 LXI H,0000
COZOMOE0I0 A=00 B=0000 D=0000
H=0000 S=10FE P=0103 DAD SP
```

```
COZOMOE0I0 A=00 B=0000 D=0000
H=10FE S=10FE P=0104 SHLD 0215
COZOMOE0I0 A=00 B=0000 D=0000
H=10FE S=10FE P=0107 LXI SP,0215
COZOMOE0I0 A=00 B=0000 D=0000
H=10FE S=0215 P=010A MVI B,05
COZOMOE0I0 A=00 B=0500 D=0000
H=10FE S=0215 P=010C DCR B
COZOMOE0I1 A=00 B=0400 D=0000
H=10FE S=0215 P=010D JNZ 010C
COZOMOE0I1 A=00 B=0400 D=0000
H=10FE S=0215 P=010C DCR B
COZOMOE1I1 A=00 B=0300 D=0000
H=10FE S=0215 P=010D JNZ 010C
COZOMOE1I1 A=00 B=0300 D=0000
H=10FE S=0215 P=010C DCR B
COZOMOE0I1 A=00 B=0200 D=0000
H=10FE S=0215 P=010D JNZ 010C
COZOMOE0I1 A=00 B=0200 D=0000
H=10FE S=0215 P=010C DCR B
COZOMOE0I1 A=00 B=0100 D=0000
H=10FE S=0215 P=010D JNZ 010C
COZOMOE0I1 A=00 B=0100 D=0000
H=10FE S=0215 P=010C DCR B
COZ1M0E1I1 A=00 B=0000 D=0000
H=10FE S=0215 P=010D JNZ 010C
COZ1M0E1I1 A=00 B=0000 D=0000
H=10FE S=0215 P=0110 LHLD 0215
COZ1M0E1I1 A=00 B=0000 D=0000
H=10FE S=0215 P=0113 SPHL
COZ1M0E1I1 A=00 B=0000 D=0000
H=10FE S=10FE P=0114 RET
COZ1M0E1I1 A=00 B=0000 D=0000
H=10FE S=1100 P=1006 RST 07
COZ1M0E1I1 A=00 B=0000 D=0000
H=10FE S=1100 P=1006 RST 07
COZ1M0E1I1 A=00 B=0000 D=0000
H=10FE S=1100 P=1006 RST 07*1006
```

```
-^C
```

Let me explain: CP/M .COM files fall into two major categories: those using all available memory which "warm boot" when they are done, and those that don't overlay CCP, but rather return to it when they complete execution.

When a routine that warm boots is being debugged, no special techniques need be considered.

However, when debugging a routine expecting to return to CCP, neither DDT nor SID set up any particular stack or return address for the program. So I created a "quickie" routine to load the stack and supply a valid return address to the program at 100. I placed the routine at 1000H where I knew there was open memory. I loaded the stack with 1100H, so there are slightly fewer than 256 bytes of stack between my quickie program and the stack. I did this as follows (where "-" is the DDT prompt character):

```
-a1000
1000 lxi sp,1100
1003 call 100
1006 rst 7
1007
```

(RST 7 is a general purpose return to DDT or SID.)

I then used the DDT "x" command to examine the program counter, and set it to 1000 where my three-line routine to load the stack and call 100 is located:

located:

```
-xp
P=0100 1000
```

DDT typed the "P=0100", and I typed the 1000 and pressed return. Now I am ready to trace the program. I decided to trace it completely, so I used a count of "ffff", or 65535 decimal.

You can follow the trace mainly by looking at the part that says "P=xxxx", for that is the Program counter. Where P=010D, you can see the instruction "JNZ 010C". That means 010C must be the label "charlp". Looking at the trace, you can see that P had a value of 10C five times, showing that charlp was executed five times.

You can also see the count in B being decremented, as the field "B=nnnn" goes from 0500 down to 0000. (In case it is not obvious, in 0500, 05 is what is in B, and 00 is what is in C. Thus B=0500 really means "BC=0500".)

Note how the trace ended with multiple RST 7's. RST 7 is the instruction used to return to DDT or SID after your program has run, or at any time you want to stop execution but still stay in DDT or SID. You can even code intentional RST 7 instructions in your assembly source, providing you only run the program under a debugger. When the "bugs are out", remove the RST 7 instructions before making a normal .COM file out of it. Placing the RST 7 instructions directly in the ASM file saves you the problem of figuring out where to place them when actually running DDT. For example, place an RST 7 where your program finishes its first task. DDT will tell you what address it was at, by typing "\*" and then the address: "\*0123". You need only "g124" to continue executing, assuming you use "x" (examine registers) and/or "d" (dump memory) to see how things look after the beginning of the program.

Back to the tracing: the RST 7 trace would have gone on for thousands of lines, were it not for the fact that I pressed return on the console. DDT is always scanning the console to see whether you want to interrupt execution during tracing.

## Breakpoints

Breakpoints furnish another way to debug a program. With breakpoints, the program executes at full speed until a certain address is reached, then it stops completely.

For example, I could have traced a couple of lines and then decided to use the DDT "go" command "g,nnnn", which executes full speed up to address nnnn. Assume I have set up my three-line piece at 1000H, and am ready to debug. To orient myself, I first list the program with the DDT "L" command. The NOP at 115 is just "garbage" left in memory.

```
-l100
0100 LXI H,0000
0103 DAD SP
0104 SHLD 0215
0107 LXI SP,0215
010A MVI B,05
010C DCR B
```

```
010D JNZ 010C
0110 LHL 0215
0113 SPHL
0114 RET
0115 NOP
```

```
-g1000,10c
*010C
```

```
-x
COZOMOE0I0 A=00 B=0500 D=0000
H=10FE S=0215 P=010C DCR B
```

```
-g,110
*0110
```

```
-x
COZ1MOE1I1 A=00 B=0000 D=0000
H=10FE S=0215 P=0110 LHL 0215
```

```
-g
*1006
```

-

The command "g1000,10c" says to go, starting at 1000h, and breaking execution at 10c. When execution reaches 010C, DDT echoes "\*010C" to show that the breakpoint has been reached. At that time, I issue the "x" command to see the registers, and get:

```
COZOMOE0I0 A=00 B=0500 D=0000
H=10FE S=0215 P=010C DCR B
```

You can see that B now contains the 05. It is still 05, so observe that the DCR B has not yet been executed.

I then decide to continue running until 110. I don't have to say "g10c,110" because DDT remembers where I was running. I can just omit the first address and type: "g,110" to go, with a breakpoint set at 110. DDT acknowledged getting to 110 by responding "\*0110". Again, I looked at the registers with "x":

```
COZ1MOE1I1 A=00 B=0000 D=0000
H=10FE S=0215 P=0110 LHL 0215
```

You can now see that B has been decremented to 0, and I am about to restore the stack pointer by first loading it to HL from 215H. I issue a final "g", which then runs until the RST 7 is hit, and DDT tells me of this by typing "\*1006". I then typed ↑C to return to CP/M.

## Patching

DDT and SID have simple assemblers built into them. But working with them is not like using an editor and assembler, because you cannot move parts of your program down to make room for inserted instructions. However, you can overlay existing instructions, or patch out unnecessary ones with NOPs, (00H).

Suppose that at 10A you wanted to load B with a 6 instead of

(continued next page)

a 5. Just type "a105" and press return. DDT responds:

```
0105
```

leaving the cursor after the 105. Then type "mvi b,6", and press return. DDT prompts:

```
0107
```

so just press return, to end the assembly mode. The program is now patched, and can be run.

If you have to insert some code, it is best to go back to the assembler and edit in the changes. If, however, the program is particularly large, DDT might still be the best bet.

At the area in your program where you want to make the changes, just put a JMP or CALL to some address outside of the program, perhaps up in high memory above CP/M, or at, say, 8000H, if you are sure your program will not use that area. If you JMP out, you will have to JMP back. If you CALL out, you can RET back. For example, suppose you have a program that under the L command of DDT looks like:

```
-l1253
1253 LXI H,0305
1256 MVI B,0B
1258 ADD M
1259 INX H
125A DCR B
125B JNZ 1258
125E RET
```

but you realize that in the routine which is supposed to add up the 11 (0B hex) bytes starting at 0305, you forgot to zero the accumulator before doing the first add. You can't insert even the one-byte XRA A instruction to zero the accumulator, so you will have to resort to going out to a patch area. First, see where DDT and CP/M-80 are, by typing "l5 7". This will show the end of the area your program may use, and specifically points to the bottom of DDT. You can use memory under that, providing you know that your program is not using that memory:

```
-l5 7
0005 JMP 9700
0008
```

I have memory open to 9700, which may seem low, but I am running under MicroShell (see *acknowledgement* at the end of this article). I choose 8000 for my patch area. Since the LXI H,0305 is a three-byte instruction, I can conveniently overlay it with a call to my patch area:

```
-a1253
1253 call 8000
1256
```

Then, at 8000, I must remember to execute the overlaid instruction:

```
-a8000
8000 lxi h,305
```

then patch in the missing instruction:

```
8003 xra a
```

and finally, return from the patch:

```
8004 ret
```

## OH-OH

Patching is very nearly a sin. Just remember to change the *source* of your program to agree with what you've patched. I have cursed myself unmercifully after fixing something with a patch, then forgetting to change the source. I have even gone so far as to use RESOURCE, my disassembler, to recover heavily patched code when I've failed to update the ASM source (or at least, when the properly modified source has been lost).

## SID

Let me re-execute the DDT trace of my test program with the symbolic instruction debugger, my favorite. You can see how much easier things are with it. To take full advantage of its features, you need a symbol table for the program being debugged. Digital Research's MAC produces the symbol table. In a pinch, you could create one yourself, since the format is simply an ASCII file with lines of:

```
nnnn label
```

in it, i.e. a 4-digit hex value, a space, then an ASCII label.

Let's look at the SID run. I traced the program, and also ran it with breakpoints. This time, the *labels* showed up.

```
SIDXREF 1D00
SYMBOLS
NEXT PC END
0180 0100 8ED8
```

Please excuse the strange name I have SID sign on with – that is to remind me that I have a little memory address cross-reference program tacked on the end of SID, so it is at 1D00. I can move it down to 100, and load the program to be cross-referenced in at 200.

I assemble my little stack-setting program at 1000:

```
#a1000
1000 lxi sp,1100
1003 call 100
1006 rst 7
1007
```

I then set the program counter to it:

```
#xp
P=0100 1000
```

then begin the tracing of my program. Again, due to the long lines, I stuck a carriage return between the printing of the D and H registers. SID, instead of commenting the PSW bits,



simply prints a "-" for bits that are off, or a single character (such as C for carry) when they are on. This was done because of the extra width of the display due to the labels being appended to the operands.

```
#tff
----- A=00 B=0000 D=0000
H=0000 S=0100 P=1000 LXI SP,1100
----- A=00 B=0000 D=0000
H=0000 S=1100 P=1003 CALL 0100
----- A=00 B=0000 D=0000
H=0000 S=10FE P=0100 LXI H,0000
----- A=00 B=0000 D=0000
H=0000 S=10FE P=0103 DAD SP
----- A=00 B=0000 D=0000
H=10FE S=10FE P=0104 SHLD 0215 .STACK
----- A=00 B=0000 D=0000
H=10FE S=10FE P=0107 LXI SP,0215 .STACK
----- A=00 B=0000 D=0000
H=10FE S=0215 P=010A MVI B,05
CHARLP:
----- A=00 B=0500 D=0000
H=10FE S=0215 P=010C DCR B
-----I A=00 B=0400 D=0000
H=10FE S=0215 P=010D JNZ 010C .CHARLP
CHARLP:
-----I A=00 B=0400 D=0000
H=10FE S=0215 P=010C DCR B
---EI A=00 B=0300 D=0000
H=10FE S=0215 P=010D JNZ 010C .CHARLP
CHARLP:
---EI A=00 B=0300 D=0000
H=10FE S=0215 P=010C DCR B
-----I A=00 B=0200 D=0000
H=10FE S=0215 P=010D JNZ 010C .CHARLP
CHARLP:
-----I A=00 B=0200 D=0000
H=10FE S=0215 P=010C DCR B
-----I A=00 B=0100 D=0000
H=10FE S=0215 P=010D JNZ 010C .CHARLP
CHARLP:
-----I A=00 B=0100 D=0000
H=10FE S=0215 P=010C DCR B
-Z-EI A=00 B=0000 D=0000
H=10FE S=0215 P=010D JNZ 010C .CHARLP
FINI:
-Z-EI A=00 B=0000 D=0000
H=10FE S=0215 P=0110 LHL D 0215 .STACK
-Z-EI A=00 B=0000 D=0000
H=10FE S=0215 P=0113 SPHL
-Z-EI A=00 B=0000 D=0000
H=10FE S=10FE P=0114 RET
-Z-EI A=00 B=0000 D=0000
H=10FE S=1100 P=1006 RST 07
-Z-EI A=00 B=0000 D=0000
H=10FE S=1100 P=1006 RST 07
-Z-EI A=00 B=0000 D=0000
H=10FE S=1100 P=1006 RST 07
*1006
```

Again, the RST 7 was executing over and over until I pressed return to stop it.

Now, let's re-execute the program with breakpoints. This time I can set the breakpoints by *label*, rather than having to know the address:

```
#g1000,.charlp
```

SID executes, and says when the breakpoint is reached, even telling me the label:

```
*010C .CHARLP
```

I examine my registers:

```
#x
-Z-EI A=00 B=0500 D=0000
H=10FE S=0215 P=010C DCR B
```

and finally go at full speed, setting a breakpoint at label FINI.

```
#g,.fini
*0110 .FINI
```

Another check of the registers:

```
#x
-Z-EI A=00 B=0000 D=0000
H=10FE S=0215 P=0110 LHL D 0215 .STACK
```

and a final "GO" to finish execution:

```
#g
```

to which SID responds

```
*1006
```

when the final RST 7 is hit at 1006. Then I use ↑C to get out of the debugger.

## Conclusions

I hope I have given you a little insight into some of the popular debugging techniques, and shown you why SID is my favorite.

Next month, I'll introduce you to MACROS - time savers for any assembler programmer.

I would like to be overwhelmed by comments, criticisms, or questions relating to this tutorial, 8080 programming, or CP/M interfacing, and I'd appreciate hearing from you c/o *Lifelines/The Software Magazine*, 1651 Third Ave., New York, N.Y. 10028.

## Acknowledgement

I could not easily have illustrated this section of the tutorial without the help of a product called *MicroShell* from New Generation Systems, Inc. 2153 Golf Course Drive, Reston VA 22091.

Of its many nice features, the one that made this article practical is "I/O Redirection", i.e. the ability to take what would normally come to the console and direct it to a disk file. This feature is patterned after the Bell Labs UNIX operating system. For example, the first DDT run began by typing "sh" to get started under MicroShell, then typing:

```
ddt test.com > +foo
```

(continued next page)

">foo" would mean to direct console output only to a file "foo", so I wouldn't be able to see what was going on. That would be appropriate for "stat >foo", where you don't need to see what you got. However, the "+" means also that the output should be directed to the console. Therefore, only the "A> ddt temp.com" had to be faked up since MicroShell only captures the output of a single program execution, and not all

console output.

The only drawback I can see is the additional memory MicroShell requires. But, "TANSTAAFL" - there ain't no such thing as a free lunch". Only when running a big WordStar edit do I really need the memory, and have to exit from MicroShell. ■

## Feature

# RMS, A Review

Davis A. Foulger

When discussing database management systems, it is easy to talk about abstract, theoretical "models of the data". But although useful, the expression is somewhat misleading. Pure applications of models of data are difficult to implement and hard to find. T.I.M. III (see the September, 1982 issue of *Lifelines/The Software Magazine*), for instance, has some features of a network model, but can only be used hierarchically. Condor, on the other hand, allows many relational operations to be performed on data which is organized around network principles. Most true database management systems for microcomputers are mixed models. Network principles are frequently used for organizing the data, but hierarchical or relational principles are most frequently used for processing it.

RMS, a database management system developed by Washington Computer Services (3028 Silvern Lane, Bellingham, WA 98226) deviates from this pattern. RMS organizes data along the lines of a hierarchical model, processes data along the lines of a hierarchical model, and offers the user only the options of a hierarchical model.

## The Organization of the Data

RMS builds data files in a fashion unlike other databases. Like all true database management systems, RMS supports more than one level of related record, allowing one-to-many relationships to be drawn within the data. It supports two levels only, not unusual among current microcomputer database systems. Unlike most of its competitors, however, RMS stores both levels in a single integrated data file. Most DBMS keep different kinds of related records in different files, relating them to one another as necessary.

There are advantages to each approach. The major advantage in maintaining separate files for each data type is the flexibility it gives the user for relating files in many different ways. It is also easier to engineer the kind of hierarchical reversal of primary and secondary file possible in T.I.M. III.

A single, integrated data file containing both primary and secondary records provides the advantage of easy use and updating and makes mistakes in entering the relationship between primary and secondary files unlikely. The nature of the relationship between differing records is more apparent to the user.

## Limitations and Applications

The structure of the data in RMS tells the whole story. The single integrated data file which the system is built around will be highly appropriate for a certain range of applications and highly inappropriate for many others. The limitations imposed by the system are simple and straightforward:

- a single primary record cannot be related to more than one kind of secondary record;

- it is not possible to select primary records on secondary fields;

- the problem of storage space for the data file will be encountered more often, and will be difficult to overcome by appeal to additional secondary files;

- time dated secondary records (as would be kept in a general ledger) are difficult to maintain;

These limitations grow more important as the number of primary and secondary records grows and as the complexity of the user's secondary record needs change. RMS can handle some complex applications, but only at a tremendous cost in storage space. Since each data file must contain both the primary records and the secondary records, complex applications will almost inevitably require the [otherwise unnecessary] duplication of records within two or more files.

RMS is best applied in applications where a limited number of primary records will be associated with many of a single secondary record type. It can be stretched beyond that range if the user is willing to do a bit of BASIC language programming, but it won't stretch far.

## Not An Easy Package

The root of most of RMS' problems in this area lies in its lack of integration. RMS has been implemented as a set of five distinct programs, each of which are run individually as needed. This lack of integration is strange, as the programming necessary to integrate the five programs into a cohesive whole seems almost trivial compared to the difficulty of writing the five modules that RMS does include.

**TABLE 1  
Facts & Figures**

<b>Package:</b> RMS (Record Management System)
<b>Price:</b> \$395 for IBM PC (also available for Z80, 6502 and others)
<b>Systems Available For:</b> IBM Personal Computer, CP/M-80
<b>Required Supporting Software:</b> Operating System
<b>Memory Requirements:</b> 48K RAM
<b>Diskette Capacity Required:</b> One Disk Drive
<b>Utility Programs Provided:</b> None
<b>Record Size &amp; Type Limits:</b> Internal storage is fixed length ASCII stored using hash coding (handle with care).  Up to fifty fixed length fields of up to 250 characters each can be stored in up to 65,535 records per file.
<b>User Skill Level Required:</b> It may take several days or weeks of trial and error programming with RMS for a novice user to master the system. Users who are experienced in data base concepts may be able to master the system in a few hours.

But even with the addition of a central program module, RMS would not be complete. Specification of data files and reports must be done with a separate text editor. Data files generated by other programs can only be utilized if the user writes a program to convert the data into RMS internal format. This last problem is a serious one, for even though the BASIC code necessary for building such a program appears in the final chapter of the RMS manual, there are many warnings against playing with RMS data files. Those warnings are sufficient to guarantee that most users won't make the attempt.

Most potential RMS users should own some form of text editor already, even if it is only the elementary line editor of SB-86 (EDLIN). Using that text editor to generate specifications for RMS' datafiles and reports should be no hardship for users, once they figure out the format for those specifications. Still, a more friendly system would have built-in utilities for generating report and data file formats and there would be no need to learn a detailed specification language.

Ultimately, these features make the RMS manual a very important component of the system. Indeed, the user's very first act will be to build a datafile specification using the manual as

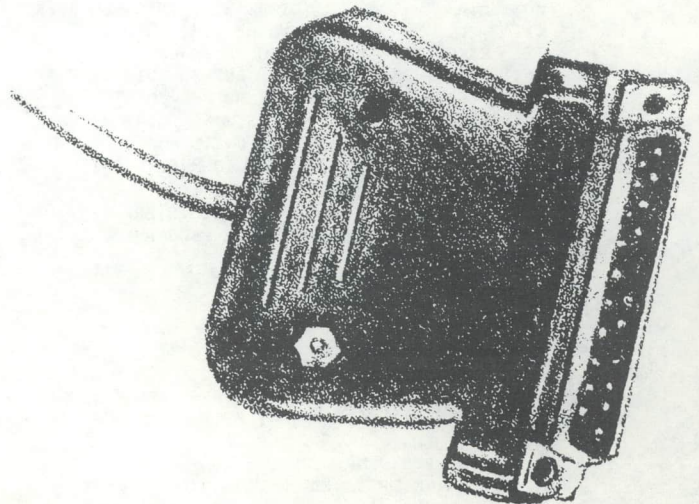
a guide to making specifications in a text editor. Fortunately the manual is pretty good, containing both reference and tutorial material.

## Conclusion

RMS is not this author's conception of an ideal database management package, but it is a very interesting package. If you can get by its unfriendly facade, RMS emerges as a rather unusual entity, with a pure hierarchical structure that makes it more appropriate to some tasks than any similar system I have encountered. I would not personally make much use of RMS unless I was able to take the time to repackage it into a friendlier and more complete whole. At \$395, however, RMS seems a little expensive for a program that will need major modifications.

**TABLE 2  
Qualitative Factors**

	Rating
<b>Documentation</b> organization for learning organization for reference readability includes all needed information	 4 4 5 5
<b>Ease of Use</b> initial start up conversion of external data application implementation operator use	 1 1 4 4
<b>Error recovery</b> from input error restart from interruption from data media damage	 4 4 1
* Ratings in this table will be in a 1-7 scale where: 1 = clearly unacceptable for normal use 4 = good enough to serve for most situations 7 = excellent, powerful, or very easy depending on the category	



(continued next page)

**TABLE 3**  
**Data Management Capabilities**

**A. Underlying Data Model**

1. *Data Types*  
Alphanumeric, Numeric, Money and Date fields.
2. *Relationships*  
Purely hierarchical supporting one to many relationships.

**B. Functions Provided**

- 1.a. *Data dictionary maintenance:*  
There is no central data dictionary and data is not file independent.
- b. *Data reorganization and conversion:*  
Short files with common formats can be concatenated into longer ones.

2.a. *Data Entry and Editing:*

The only part of the package that is really easy to use, with good facilities for both entry and editing of data. Speed is relatively good, although form generation is rather inflexible and simplistic.

b. *Report Generation:*

Reports are generated from a single file containing both primary and secondary records. Totals, subtotals, and group subtotals can be computed.

3.a. *Data selection by predicate:*

Not Available.

b. *Data Joining and Relating Multiple Data Sets:*

No.

c. *Calculations on Data:*

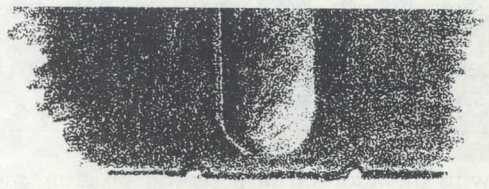
Totaling can be performed in reports.

4.a. *Data Independent Application Interface:*

None.

U.S. Postal Service STATEMENT OF OWNERSHIP, MANAGEMENT AND CIRCULATION <small>Required by 39 U.S.C. 3685</small>			
1A. TITLE OF PUBLICATION LIFELINES		1B. PUBLICATION NO. 2. DATE OF FILING 10-8-82	
3. FREQUENCY OF ISSUE MONTHLY		3A. NO. OF ISSUES PUBLISHED ANNUALLY 12	
4. COMPLETE MAILING ADDRESS OF KNOWN OFFICE OF PUBLICATION (Street, City, County, State and ZIP Code) (Not printers)		3B. ANNUAL SUBSCRIPTION PRICE \$24 US, \$50 F	
1651 Third Avenue, New York, NY 10028			
5. COMPLETE MAILING ADDRESS OF THE HEADQUARTERS OF GENERAL BUSINESS OFFICES OF THE PUBLISHER (Not printers)			
1651 Third Avenue, New York, NY 10028			
6. FULL NAMES AND COMPLETE MAILING ADDRESS OF PUBLISHER, EDITOR, AND MANAGING EDITOR (This item MUST NOT be blank)			
PUBLISHER (Name and Complete Mailing Address) Intersoft Corporation, 1651 Third Avenue, New York, NY 10028			
EDITOR (Name and Complete Mailing Address) Dr. Edward H. Currie, 1651 Third Avenue, New York, NY 10028			
MANAGING EDITOR (Name and Complete Mailing Address) Jane V. Mellin, 1651 Third Avenue, New York, NY 10028			
7. OWNER (If owned by a corporation, its name and address must be stated and also immediately thereunder the names and addresses of stockholders owning or holding 1 percent or more of total amount of stock. If not owned by a corporation, the names and addresses of the individual owners must be given. If owned by a partnership or other unincorporated firm, its name and address, as well as that of each individual must be given. If the publication is published by a nonprofit organization, its name and address must be stated.) (Item must be completed.)			
FULL NAME		COMPLETE MAILING ADDRESS	
Intersoft Corporation		1651 Third Avenue, New York, NY 10028	
Oak Management Corporation		2 Railroad Place, Westport, CT 06880	
Bessemer Venture Partners		630 Fifth Avenue, New York, NY 10111	
Larry B. Alkoff		1651 Third Avenue, New York, NY 10028	
Anthony R. Gold		1651 Third Avenue, New York, NY 10028	
8. KNOWN BONDHOLDERS, MORTGAGEES, AND OTHER SECURITY HOLDERS OWNING OR HOLDING 1 PERCENT OR MORE OF TOTAL AMOUNT OF BONDS, MORTGAGES OR OTHER SECURITIES (If there are none, so state)			
FULL NAME		COMPLETE MAILING ADDRESS	
9. FOR COMPLETION BY NONPROFIT ORGANIZATIONS AUTHORIZED TO MAIL AT SPECIAL RATES (Section 423.12 DMM only) The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes (Check one)			
<input type="checkbox"/> (1) HAS NOT CHANGED DURING PRECEDING 12 MONTHS		<input type="checkbox"/> (2) HAS CHANGED DURING PRECEDING 12 MONTHS (If changed, publisher must submit explanation of change with this statement.)	
10. EXTENT AND NATURE OF CIRCULATION			
		AVERAGE NO. COPIES EACH ISSUE DURING PRECEDING 12 MONTHS	ACTUAL NO. COPIES OF SINGLE ISSUE PUBLISHED NEAREST TO FILING DATE
A. TOTAL NO. COPIES (Net Press Run)		12,500	15,000
B. PAID CIRCULATION			
1. Sales through dealers and carriers, street vendors and counter sales		1,100	1,420
2. Mail Subscription		7,980	9,303
C. TOTAL PAID CIRCULATION (Sum of 10B1 and 10B2)		9,080	10,723
D. FREE DISTRIBUTION BY MAIL, CARRIER OR OTHER MEANS SAMPLES, COMPLIMENTARY, AND OTHER FREE COPIES		240	25
E. TOTAL DISTRIBUTION (Sum of C and D)		9,220	10,748
F. COPIES NOT DISTRIBUTED			
1. Office use, left over, unaccounted, spoiled after printing		3,133	4,252
2. Return from News Agents		47	0
G. TOTAL (Sum of E, F1 and 2—should equal net press run shown in A)		12,500	15,000
11. I certify that the statements made by me above are correct and complete		SIGNATURE AND TITLE OF EDITOR, PUBLISHER, BUSINESS MANAGER, OR OWNER <i>E. Currie</i> Chief Operating Officer	

PS Form 3526  
July 1982



(continued from page 7)

```

MOVPGM:  MOV A,M           ;GET BYTE FROM STRING
          INX H            ;BUMP POINTER
          SHLD STRPTR      ;SAVE POINTER
          LHLD BUFPTR      ;GET BUFFER POINTER
          MOV M,A          ;STORE BYTE IN CNSLE BFFR
          CPI NULL         ;END OF STRING ?
          JZ EXECUT        ;EXECUTE PROGRAM
          INX H            ;BUMP BUFFER POINTER
          SHLD BUFPTR      ;SAVE BUFFER POINTER
          LHLD STRPTR      ;GET STRING POINTER
          JMP MOVPGM       ;LOOP UNTIL DONE
EXECUT:   MVI A,08H        ;SETUP COMMAND POINTER
          STA CMDPTR       ;STORE IT
          MOV C,DRIVA      ;INTO C REGISTER
          JMP TRADDR       ;EXECUTE PROGRAM
;*****
;*
;* MENU SCREEN DISPLAY *
;*
;*****
MENUSCR: DB CLRSCL, NULL, NULL, ' SAMPLE MENU PROGRAM'
          DB CL, LF, '-----', CR, LF, LF
          DB LF, '1. RUN PIP.', CR, LF, LF
          DB '2. RUN STAT.', CR, LF, LF
          DB '3. RUN DUMP.', CR, LF, LF, LF
          DB 'ENTER OPTION: #', BS, '$'
    
```

```

;*****
;*
;* EXECUTE STRING TABLE *
;*
;*****
PROG1:DB ((PROG2-2)-$), 'PIP', NULL
PROG2:DB ((PROG3-2)-$), 'STAT', NULL
PROG3:DB ((PGMTABL-2)-$), 'DUMP STAT.COM', NULL
;*****
;*
;* EXECUTE STRING START *
;* ADDRESS TABLE *
;*
;*****
PGMTABL:EQU $ ;TABLE OF PROGRAMS
P1: DW PROG1 ;POINTER IS MENU
P2: DW PROG2 ;OPTION
P3: DW PROG3
PTLNGLTH:EQU $-PGMTABL
MAXOPT EQU PTLNGLTH/2 ;HIGHEST OPTION
;*****
;*
;* STORAGE DATA AREAS *
;*
;*****
STRPTR: DW 0
BUFPTR: DW CONBUF
;
END
    
```



Now for the most significant shortcomings. These are functions which some of the other programs perform better:

- 1) WordStar is one of the most difficult of all the CP/M word programs to learn. Once you know it, of course, it gets easier, but in many businesses the people who have to run the word processor come and go, or the need arises to call in temporary help for overflow work, illnesses, and so on. Even in my own little one-man operation, I sometimes want someone else to come in to type a big manuscript into the computer so I can edit it. In cases like this, WordStar is not the best choice.
- 2) Even when you learn WordStar, it is still harder to use than some of the other programs. The command structure is too complicated, there are too many commands to remember, and too many keystrokes are required. It is just plain awkward. Any user, especially people who write a lot, should have an editor which very quickly becomes transparent - i.e., you forget about the program and just do it. Nothing should distract your attention, or come between you and your thoughts on the screen. The word processor should be like playing a chord organ, whereas WordStar is more like mastering the harp, where you have to concentrate and pick your way through it.
- 3) WordStar's "include" functions are inadequate. Being able to display an external file to look up information is extremely important for almost any kind of work, as is the ability to insert any selected portion of an external file into the text in RAM, at any point indicated by the cursor. You should also be able to call in portions of a boilerplate file by code number. Once you have worked with a program that offers these features, you will never want to do without them. Unfortunately, all of this is either clumsy or impossible in WordStar. You cannot look at any file other than the one in RAM. You cannot include a portion of an external file; instead, if you need material from another file you have to load the whole thing in, then delete unwanted portions. Small files can be put in at any cursor location and dealt with, but with larger files you

tend to load the whole thing at the end, amputate, and block move the selected portion to where you need it. You can't call in coded portions of a boilerplate file; the only way to do this in WordStar is to make a separate file of each segment. Clumsy. Messy. Imagine what the directory looks like for someone who has to do a lot of boilerplating.

- 4) WordStar fails to take advantage of terminal features, such as cursor-keys, and special function keys. Since most other programs use the cursor keys, this makes using WordStar frustrating - you have to be reminded with glitches and errors to keep your fingers off the cursor keys.
- 5) Mail merge functions are not included, and to get them you have to pay extra for MailMerge.
- 6) Very poor (i.e., no) protection and recovery from disk-full error, which may cause you to lose work, and will certainly bog down your activity. This problem is caused by the buffered disk operation which permits you to work on long files, but the problem can best be avoided by *not* working on long files. Catch 22. Innocent edit activities - such as long cursor moves through a big file, global search/replace, and especially block manipulations - can have unexpectedly troublesome results. To avoid problems, the user is required to constantly monitor disk space, trying to keep twice as much empty space as the longest file. This is expensive for people who have lots of long files, and bloody inconvenient for people with limited disk space. WordStar should (but does not) automatically protect you against disk-full problems, or at the very least should give you tools to cope, perhaps a command which would show disk space without leaving the edit mode.
- 7) More clumsiness. I hate having to reform paragraphs, especially given that I like to work and rework my documents. This is the kind of housekeeping that computers should handle, and I shouldn't have to. In fact, I don't know off hand of any other good program that makes you do anything like this.

- 8) Some users desire or require the highest possible quality of print for making books, reports, newsletters, catalogues, brochures, and so on. WordStar does not support the proportional print capabilities of the specialty printers, and it does not support the built-in special features of most dot-matrix printers, instead treating them like simple teletype devices.

It began to seem even to me that my opinions on WordStar are so strong as to be suspect. Maybe I'm just prejudiced or eccentric. In an admittedly simplistic effort to test this possibility, I called on an acquaintance for an independent second opinion. My friend is an electronic engineer turned programmer who has put years in on monster machines and high-level programming. He recently entered the micro field, hoping to develop an independent income, and his first word processor is, of course, WordStar. Jim wrote: "I don't care how good WordStar is technically, it's human engineering is totally unacceptable! To think that two software companies and several authors can make a living providing band-aids for MicroPro's ineptness is revealing of the immaturity of the microcomputer software market. I would *never* turn this program over to a novice."

As you see, he reacts even more strongly than I do, but his point about the authors and publishers is well taken. Several books are published (and apparently selling quite well) showing how to use WordStar. This indicates a widely distributed but hard to use program. A good word processor does not need such support. The software companies he refers to publish customization aids for WordStar in effort to patch its shortcomings, which brings us to the next section of this column.

## The Big Fix

For those of you who already have and will continue to use WordStar, and for those of you who are completely unimpressed by my carping, there is a worthy program you should know about which goes a long way toward making WordStar easier to learn and use.

Here's how it works. Take one Word-

Star program, add *Quickey*, and, Shazam! You now have a formidable word processor with single-stroke function keys all out on the keyboard where they belong. The innards are still WordStar, so it still lacks the ability to display or to conveniently include portions of external files, it still won't support proportional printing, and it still has poor protection against disk-full errors. But suddenly WordStar is one heck of a lot easier to live with, and more productive.

*Quickey*, published by Raish Enterprises of Levittown, N.Y., is a completely customized reworking of the "front end" of WordStar. *Quickey* rewrites the command structure of WordStar, making it more logical, more streamlined, and more mnemonic. Then it puts almost all commands, forty-five of them, right out there on the keyboard where you can see them. Most commands are accomplished with a single keystroke. No more groping for crib cards and manuals. No more triple keystrokes, and almost no double-strokes. This is more like it! And they even send replacement keycaps with new labels for your keyboard. The suggested retail price is \$150.

Here's a summary of the features:

- \* All cursor keys are usable, and work in combination with (amplified by) the home key.
- \* All menu prefixes are single keystrokes. Each menu has been simplified and renamed mnemonically.
- \* All block operations are a single keystroke.
- \* Most insert and delete functions are a single keystroke.
- \* Paragraph reform, paragraph tab, and margin release are single keys.

The keyboard layout is shown in the illustration, which will give you a more complete picture. Check it out.

An example of *Quickey* innovation is found in the cursor commands. This may be the cleverest formulation around. Under *Quickey*, cursor keys have the obvious result, moving the cursor one step in the direction indicated by the arrow. The HOME key is a motion amplifier. Prefixing any cursor key with HOME moves the cursor as far on the screen as it can go in that direction. HOME is also used in

other combinations for cursor motion:

HOME HOME	cursor to top of file
HOME E	cursor to end of file
HOME n	cursor to marker "n"
HOME B	cursor to beginning of marked block
HOME /	cursor to end of marked block
HOME P	cursor to previous position
HOME S	cursor to start of last find, or to source of last block manipulated.

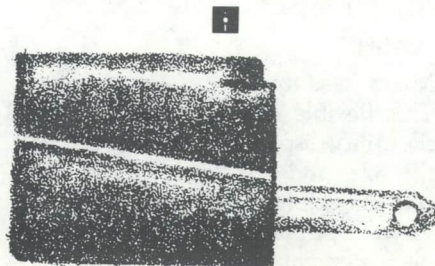
Word-right and word-left tabs each get their own function key, and so do scroll-up and scroll-down. Here again, HOME is an amplifier.

Another small irritant is that the word-left and word-right keys are used in conjunction with the shift key, and are located over the 7 and 8 keys. This forces an uncomfortable stretch for keys which some people use frequently.

Now for the good news. When I called Raish to report these wrinkles I found them to be extremely responsive, concerned, and responsible. They called back promptly and spent a transcontinental hour chewing it over. Right on the spot, Raish figured out what the problem was and how to correct it. Future *Quickeys* will have the word-left and word-right keys relocated, and any command which tends to get punched in rapid sequence will be relocated to keys which generate only two-character codes. If you have an old copy of *Quickey*, contact Raish for a fix. Now that's what I call good support. I get the impression that if you have any kind of questions or problems you'll be equally well-treated.

## Forecast

In some future column, I'll do a comparison of the other CP/M word processors, and try to relate features to selection criteria. See you next month.



## Something New

When you can't find your problem, let ACTIVE TRACE show it to you! See inside your program as it's working! Just as important, see inside your program when it's *not quite* working!

New to Basic? ACTIVE TRACE will let you see what Basic does as it does it! ACTIVE TRACE displays the line number, name, and current value of the variables and functions you choose, as they are encountered in program flow.

## Something Old

Though less exciting than harnessing the power and speed of your computer to find mistakes, using your computer to avoid mistakes in the first place is equally valuable. Cross-reference utilities have been around for a long time. Most programmers would not attempt to work without them, and we don't know why they have not become more well known and understood among Basic programmers and educators. ACTIVE TRACE produces complete cross-reference maps and explains their use and importance.

## Active Trace

If you have great intuition and are well-disciplined, then you'll want ACTIVE TRACE. But if you're like the rest of us, you need ACTIVE TRACE to:

- Understand and modify programs you did not write
- Improve your programming skills
- Minimize program development time

**\$125.00**

complete with primer to help you use ACTIVE TRACE to improve your programming.

Why pay more for cross-reference utilities alone when you can have ACTIVE TRACE, the new easy to use programming environment for the Microsoft family of Basic interpreters.

SOFTWARE  
SOFTWARE  
**DIGITAL MARKETING**  
DIGITAL MARKETING™



**DIGITAL MARKETING CORPORATION**

2670 CHERRY LANE • WALNUT CREEK • CALIFORNIA • 94596  
(415) 938-2880 • Telex 17-1852 (DIGMKTG WNCK)

ACTIVE TRACE is a Trademark of The Data Works.

# Product Status

## Reports

The new software products and new versions described below are available from their authors, computer stores, software publishers, and distributors. Information has been derived from material supplied by the authors or their agents, and *Lifelines/The Software Magazine* can assume no responsibility for its veracity. Software of interest to our readers will be tested and reviewed in depth at a later date.

## New

## Products

### CREDIMAX

Syntropy, Inc.

This package manages basic credit union accounts, processes transactions and maintains balances. It handles up to 32,000 member numbers, 256 share accounts per number, amounts up to \$9,999,999,999 and as many transactions per member as disk space permits. Up to 255 payrolls are supported.

Members and their Share Accounts, Payroll Deductions, Loans (including Line of Credit and Partial Disbursements), Payroll characteristics and Account Transactions can be reported or modified. Deductions may apply to loans, share accounts, or automatic transfers. System parameters can be set for automatic dividend generation. Batch balancing utilities permit the verification of transaction particulars before posting them to account balances. Posting features update account balances and supply period-to-date and year-to-date information. Other reports inform users as to Credit Union Status overall, Delinquent Loans, Insurance Data for CUNA reports, Balancing Totals and Posting Summary. Federal 1099 forms can be generated.

The software is priced at \$5000; a Z80, CP/M-80 2.2, 24 by 80 CRT with cursor addressing, 64K RAM, 55K TPA, 132 column printer, and hard disk storage are needed.

### Moneytrack

Pacific Data Systems

This money management package maintains transaction records for small businesses and personal accounts. It prepares reports, prints checks and helps with bank reconciliation. Reports include lists of transactions by Business/Account, Selected Business/Account, Fund and Selected Account. In addition, balances by Fund, Account/Business, and Business/Account can be reported, as can Repeating Items; an Entry Audit report is also available.

This product includes a UCSD p-system run time package and requires no operating system. Up to 99 funds and up to 900 accounts (shared by up to 99 businesses) are supported.

Moneytrack requires 64K, two 320K disks, a 24 by 80 CRT, and a printer supporting 80 characters per line. It runs on the IBM PC and is priced at \$450.

### Quic-N-Easi AG

Standard Microsystems

This applications database manager is intended for the novice and requires no programming experience. Input screens with blanks to be filled in can be created by the user. Data edits are set up on a screen form. Function key programs are selected from a table and indexed sequential file access is featured.

Validity checks and screen calculations are supported. Data fields may be added or deleted and data files are automatically regenerated by the applications generator.

The report generator employs no procedural language, and the reports are defined on screen by typing them the way they will print. The report generator can resort data so that it can be reported in any order. Up to six files from the applications generator may be used simultaneously by the report generator. The package costs \$295.

### WASH

Micro Resources

This flexible directory and CP/M-80 file utility displays the directory, checks file sizes and disk space. In addition, files may be renamed, copied, deleted, viewed, printed and tagged for multiple file operations.

WASH runs on CP/M 1.4 or 2.2 and loads at 0100H of the CP/M transient program area; it extends about 8K bytes into the TPA. During operation WASH displays a command menu. The drive specified by the command line is selected and all specified directory entries are read into a memory resident list. This list is sorted into alphabetical order and displayed on the screen. Two display modes are designed for a) clear screen/cursor addressing equipped terminals and b) hard copy or scroll only console devices.

The operator may select a specific file from the file list, by moving forward and backward through the list. A command to ZIP ahead through the list allows rapid access to any given file. The file list is treated as a circular buffer so that forward or backward scanning wraps around the list. Once a file name is selected, the file may be:

- viewed at the console (if text)
- listed at the list device
- sent to the punch device
- renamed with only the new name typed in by the user; normal file name validity is checked and data entry editing with backspace delete is supported
- deleted
- copied to another drive or to another user area of the same drive.
- The size of the selected file may be displayed in number of logical records and kilobytes.

A provision is included to selectively tag files from the file list; the tagged files may then work with copy or delete commands which allow all tagged files to be operated on. The Tagged Copy command copies files to another disk or to another user area of the same disk; the Tagged Delete command includes a file by file prompting option for confirming each delete.

Other commands permit the operator to determine space remaining on another drive, or restart WASH operation on another drive or user area of the same drive. Some commands are available only to CP/M 2.2 users. The COPY and RESTART/LOG commands under CP/M 2.2 can be set to allow file copying and restart across user boundaries when accessing another disk drive.



The WASH 3.2 installation package permits tailoring of the console interface. Any legal drive may be chosen as the file list source.

The product is implemented in 8080 assembly language and no assumptions are made about the maximum number of directory files, other than available memory space for the list. Directory and disk I/O is handled through BDOS calls, guaranteeing compatibility with all system implementations. WASH is priced at \$49.95, plus tax.

## New

---

---

---

---

---

---

---

---

## Versions

Please see pages 35 and 36 for reports new versions of CP/M-80 and dBASE II. muMATH/muSIMP, from Microsoft, Inc., is now available in a version for the IBM PC.

### Pascal/Z and Pascal B/Z

---

Version 4.1

New features include:

- 15% faster runtime
- an INCLUDE statement generated in the .SRC file to include correct MAIN, EMAIN, XMAIN or XE-MAIN during assembly
- error messages are generated if INCLUDE file is empty or not found, or if READLN and WRITELN of non-text files is disallowed
- the total number of compilation errors is reported in the .LST file and on the console
- a compiler option initializes (if enabled) local variables of a procedure or function to zero when the procedure or function is entered
- sixty overlay modules are now permitted and the user may specify an overlay starting address
- OVLGEN works under XSUB and OVLGEN-generated modules are Microsoft-compatible
- range checking is performed on pointers with R option enabled
- division of a constant by zero is detected at compile time
- a TAB is not treated as a string terminator when reading from console
- if REALs are entered incorrectly from the console the user will be re-

prompted and a fatal error will no longer occur

- RESET or REWRITE on a file name returns an error message
- Pascal/Z is MP/M compatible
- LINK/Z can be made compatible with Microsoft linkers past version 3.36
- a new version of PASOPT optimizer is included
- run-time error messages are listed in the manual

Library modules and source files, as well as one file of the fixed point package, have been modified.

If the assembler is given "y" as the listing drive, it will output the listing file to the printer, as does the compiler.

Users should remember to reassemble and relink any external routines, since the assembler and linker now accept up to eight significant characters. Ithaca Intersystems has also mentioned that the example MOD—STRING on p. 83 of the manual requires two more global types to work properly:

```
TYPE $STRING0 = STRING 0;  
$STRING255 = STRING 255;
```

Pascal B/Z is a business programmer's version of the Pascal/Z compiler; the floating point routines are replaced by BCD (binary coded decimal) fixed point routines, to allow greater precision and accuracy. Up to thirty digits of precision are supported, under user control. Fixed point numbers of different sizes can be mixed within the same program.

### Priorities

---

Version 1.1

This time management system has been updated to include more printing options and to increase speed. Overdue prioritized tasks are now reprinted on successive daily reports - as a reminder to the user. A new Multi-Day report permits reporting through setting of several values: start date, end date, search string and column format.

## Books

---

---

---

---

---

---

---

---

### Reviewed by Steve Patchen

*Problem Solving Principles For ADA Programmers: Applied Logic, Psychology and Grit*

by William E. Lewis  
Hayden Publishing Co., Rochelle Park, N.J., 1982  
\$9.95

This book approaches programming as a problem-solving technique. Three aspects of problem-solving are discussed: general, program-related and the influence of psychology on the process. Most examples are in ADA, but knowledge of ADA is not essential to understanding the terminology and techniques discussed. Different versions of the book show examples in BASIC, in PASCAL, in FORTRAN and in INTERLINGUA.

The first section of the book introduces problem-solving and thinking tools as concepts. Twenty-two general problem-solving principles called prescriptions are covered. Then the application of top-down development is related to the problem-solving process. The last section of the book applies sixteen related problem-solving prescriptions for program debugging. Each prescription is illustrated by both computer and non-computer examples.

Most prescriptions have catchy titles like: 'Make Sure There is Method to Your Madness', 'He Who Digs a Pit Will Fall into It' and 'Zeal Without Knowledge Is the Sister of Folly'. The problem-solving process is diagrammed as four steps: of problem definition, solution planning, coding and debugging. The author emphasizes that concentrating on proper analysis in the first two steps reduces the amount of time spent in backtracking, makes the remaining backtracking easier and gives assurance that the problem can be solved. The problem structure is related to the concept of input, processing, output. Thus, a problem has a set of facts given, operations and a goal to be achieved. The thinking tools are used to work forward and backward along this path between the given facts and the goal and to leap hurdles encountered along the way. The prescriptions introduce either principles of procedural logic or psychological principles dealing with the mind and creative process.

The second chapter introduces the first prescription, 'Make Sure There is Method to Your Madness'. It instructs the reader to be sure that the problem is defined completely before trying a solution. Once the definition is clear, the problem relationships can be determined and a solution planned. The discus-

(continued next page)

sion of this principle is completed by a non-computer example and two computer examples. Most of the other prescriptions are presented in a similar manner.

Typical chapters which cover psychological principles recommend that you 'Incubate When Gears Get Stuck' or that you avoid adding additional stress to the process by insisting upon a solu-

tion as soon as possible. The author also discusses what he calls the 'game effect'. Pleasure in problem-solving can spark interest in the problem and provide motivation towards a solution. However, enjoyment of the game can also interfere with the technical handling of the problem.

The debugging section treats bugs as problems to be solved. After describing

the relationships between symptoms and bugs, another set of prescriptions is presented for the solution of this type of problem. The book concludes with a bibliography of related works on problem-solving, logic and psychology.

I found this book to be easy, enlightening reading, useful to both experienced programmers and beginners. ■

## CPMUG News Ward Christensen

# CP/M Users Group

## Online Communications Between CP/M Users:

### Announcing CBBS/CPMUG, (312) 849-1132

CBBS/CPMUG (C/C for short) is a Computerized Bulletin Board System I have put up to supply online communications among people interested in CPMUG. It is available 24 hours a day, seven days a week. It is a single user system, so please be considerate about the amount of time you spend.

### Calling CBBS/CPMUG

Use any standard 103-type modem: 110, 300, 340, or 600 baud. The system is using a PMMI. Press return several times for CBBS to detect your speed. Be sure auto-linefeed is set off, or I'll never see two consecutive carriage returns.

### Overview of CBBS

CBBS was created by Randy Suess and myself in January 1978. We put up the first one (still online at 312-545-8086) in February '78. CBBS gets its name from the fact that it is patterned after the "cork board and push pin" bulletin board often found at computer club meetings, where people post 3x5 cards with notes to each other. On CBBS, the "cards" are messages - each assigned a number and each having a heading with pertinent data. An example:

```
00001 80 10/03/82 W.CHRISTENSEN ALL
      Subj:CPMUG HISTORY
```

This means "Message 1 is 80 lines on

10/03/82 from me, about CPMUG history". The S (Summary) command supplies this information. There is also a Q command for Quick summary. The command will stop when it runs out of messages, or may be suspended (to stop it scrolling off your screen) by typing control-S or just S. Restart with another S or with Q. To abort the summary and return to the main menu, hit control-K or just K.

Here is a quick summary of the low numbered messages. It was obtained by typing Q at the main function prompt then when asked for the starting message number, replying 1. Since answers to CBBS questions may be "stacked" using ";" as a delimiter, the command could have been typed: q;1

```
00001 CPMUG HISTORY
00002 CBBS/CPMUG PURPOSE
00003 CBBS/CPMUG FILE TRANSFERS
00004 MODEM DOCUMENTATION
00005 MODEM PROTOCOL (CONT'D)
00006 CPMUG CONTRIBUTION FORM
00009 CBBS HARDWARE/SOFTWARE REQMT
00010 CBBS SOFTWARE FOR SALE
00011 CBBS ORDER/ T & C FORM
00012 USER PROFILES
00060 V60 6502 Z-80 UTIL
```

Message numbers 60-nn will contain the CPMUG catalog files from the respective volumes. If there are requests, I'll put older catalogs on, too.

### Commands

Most CBBS commands are single letters, entered at the main function menu. The major functions supported are:

- (S)ummarize msgs
- (Q)uick summary
- (R)etrieve msg
- (E)nter message
- (H)ELP
- (G)ood bye

More minor functions are:

- # Print caller # etc
- (A)lter Baud rate
- (B)ulletin reprint
- (C)ase upper/lower
- (D)uplex: echo off
- (K)ill message
- (N)ulls: How many?
- (P)rompt bell off
- (T)ime/date print
- (V)ideo backspace
- (W)elcome reprint
- E(X)pert user mode

Additional commands are:

- CHAT See if operator is available to talk via keyboard
- CONT Continue previous message (use E for first part)
- HELP New user help; (H keyword based help)
- MINE Find my messages
- NEWS What's new on CBBS
- SHORT Shorten output

You can obtain details on any command with the H command, which accesses a large file of many keywords. All commands execute by typing the command and pressing return. If you then type a command, you will get help for it.

## File Transfer

I am deeply concerned that unrestricted file transfers would tie up the system, making it less available for message traffic. For the time being, I am making an area available for public use, under the password CPMUG. You have to execute the otherwise undocumented M command to get into the modem sub-menu mode. From there it supports five single-letter commands: D Directory list; M go to Main menu; R Receive file from you; S Send file to you; and T Type file. A "?" command prints the list of available commands. STAT is also available, and prints the amount of space available on the disk.

Among the files I have placed on it is Q.ASM, the ASM file I couldn't print in my 8080 programming tutorial a couple months back (you had to type in and load Q.HEX, which was a pain).

The modem protocol supported is from my original modem program from 6, 25, 40, 47, and more recent volumes.

The CRC protocol implemented in later versions is not supported. Documentation on the protocol is contained online, in messages 4 and 5.

The areas I foresee file transfer being used for are:

- Sending contributions
- Retrieving contributions for review
- Modemming in bug fixes for previous releases

I also send articles to *Lifelines/The Software Magazine* once a month, so if you want to modem something in, I can get it to them. (You'll get any \$\$ due you - or you can let me put it toward supporting CBBS/CPMUG whose [not insignificant] expense I have personally taken on.)

If you plan to contribute by modemming to CBBS/CPMUG, include a CPMUG contribution form (on most recent CPMUG disks, available as message 6 on C/C, or may be modemmed from C/C). U-G-FORM.LIB is the full form with comments and U-G-FORM.-BRF is the brief form with all instructions deleted.

## Purpose Of CBBS/CPMUG

CPMUG lacks anyone for users to "talk" to. It is not appropriate to

employ someone just to talk to people with questions about CPMUG, as the volumes could not be maintained at their current low price.

A CBBS can be a vehicle for many people. Unlike telephone communication, where only two parties benefit, a CBBS allows anyone to "look over the shoulder" at what is going on, and participate if they feel like it. The kinds of subjects already on CBBS/CPMUG as of the end of October are:

00500 ERROR IN CPMUG V.78  
00503 PROFILE: JIM WILLING  
00513 STOIC CP/MUG #23  
00522 DIR LOCATION  
00531 CP/ & ATARI 400  
00532 CPMUG OBJECT LIBRARY  
00538 OBJLIB & CPM3  
00546 STOIC HELP  
00576 CONTRIB: BILL DAVIDSEN  
00580 STARTING AN RCPM/CPMUG 91+  
00586 HELP CPMUG:WRITETOMAGAZINES

500 is a bug report; 513 a request for help on STOIC; 503 a profile of CP/M users; 522 a request for information about location of directory on CP/M disks; 531 a request for info on hooking an Atari to a CP/M machine as an intelligent graphics peripheral; 532, 3, and 8: comments on the CPMUG object library; 546 help for the STOIC question; 576 a list of contributions from Bill Davidsen; 580 information on upcoming releases; 586 a request to arrange with magazines to contribute software they publish.

These are just samples. The shape of C/C will be determined by its users. I see the primary goals as "communications among people with like interests", but more specifically, messages about:

- contributing software;
- reviewing software;
- bug reports/fixes;
- asking if a certain kind of program exists;
- "Conferences": many people leaving messages on a single subject;

Example: discussion of "CPMUG OBJECT LIBRARY" - a library of source and object routines which, using Digital Research's RMAC, will simplify writing assembly programs for the 8080.

I plan to put a record of each CPMUG contribution in a message, so all can see what is coming. Contributors can check the progress, report bugs, etc. You

can participate in the cataloging of CPMUG volumes by helping decide which contributions would nicely go together to make a volume.

C/C is also an excellent vehicle for contacting me about anything - 8080 programming tutorial, editor reviews, CPMUG contributions, etc.

I welcome your calls and ideas. If you have comments for me not of general interest to all callers, leave them when you issue the G command to say (G)ood bye, leaving messages open for more general communications.

## Ordering From CPMUG

Many of you have inquired about ordering volumes from The CP/M Users Group (CPMUG).

The complete catalog of volumes is available for \$10, prepaid, to the U.S., Canada and Mexico. The price is \$15 for catalogs sent to all other countries.

Software is obtainable exclusively on diskette; CPMUG does not supply printed documentation.

Software is available in 8" IBM, NorthStar 5¼", and APPLE 16 sector formats.

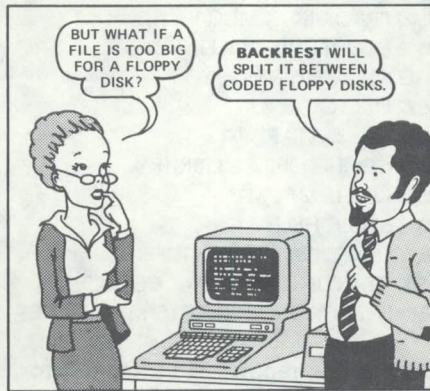
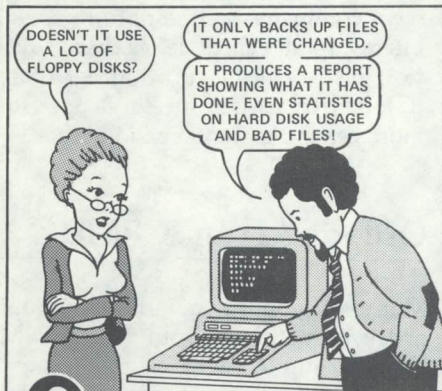
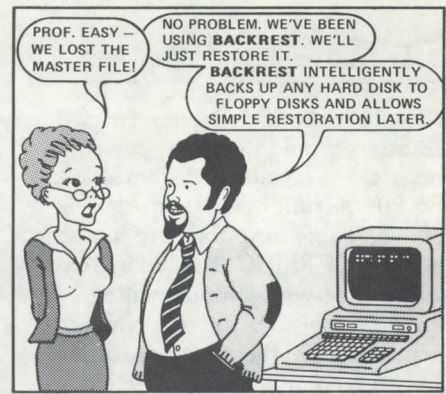
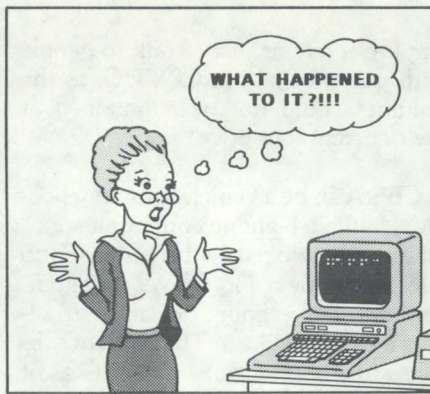
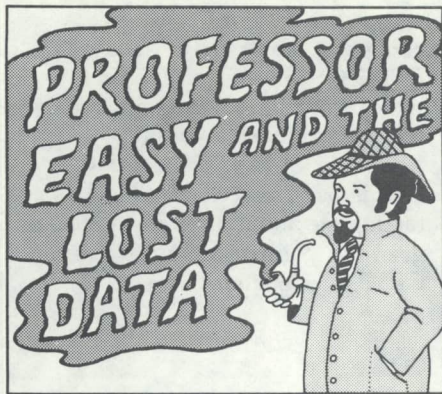
Payment covers the cost of material, packaging, and postage (which is exorbitant). Checks must be in U.S. dollars, drawn on a U.S. bank.

CPMUG receives orders by mail only; they have NO phone service (contrary to misinformation supplied occasionally by other publications such as BYTE).

Orders should be sent, with prepayment, to CPMUG, 1651 Third Ave., New York, N.Y. 10028.

## CPMUG Library Available For APPLE

In response to your demands, CPMUG has now made its library of 90 volumes of public domain software available to users of APPLE, 16 sector diskettes. ■



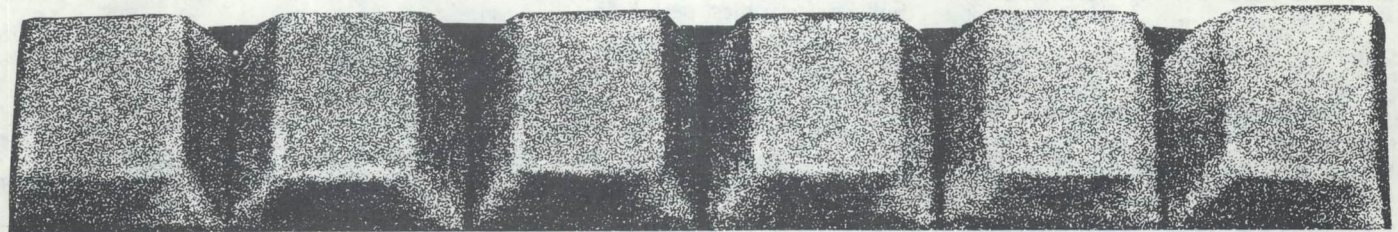
**Si** Stok Software Inc.  
 17 West 17th Street  
 New York, N.Y. 10011  
 (212) 243-1444

Complete 8 inch CP/M format disk and manual retails for \$99.95. N.Y. residents please add sales tax.

Toll free order line: (800) 431-1953 ext 183  
 In NY (800) 942-1935 ext 183



Dealer inquiries invited.  
 CP/M is TM of Digital Research



## Software Notes

# Once More with Feeling: CP/M Version 3

Michael Olfe

### Introduction

Once upon a time, when people spent hours tuning the volume and treble controls of their cassette recorders so that BASIC would load, when grown men had knobs on their fingertips from flipping front panel switches, and when software came punched on paper tape, I saw an article in *Dr. Dobbs Journal* titled "First word on a floppy-disk operating system". The subtitle was "Command language and facilities similar to DECSYSTEM-10". The price was (tentatively) \$50 for the documentation, and \$20 for the software. Some months later, after a to-the-death struggle with my Tarbell disk controller, I saw 'A>' for the first time.

Then came version 2.2, and not only did all the filenames stay

on the screen when you typed "DIR", but you could actually put more than 241K on an 8" disk! They'd gone about as far as they could go, one thought.

Then all sorts of wonders befell us. Hard disks, 16-bit CPUs, memory management, multi-tasking, multi-users, parallel processing, etc. The software started to look nice, too. When IBM stopped laughing at microcomputers and began to sell them, it was clear that the world would never be the same.

So here is CP/M again, version 3. I'm a little sad that I can't feel the same excitement about this event that I did about seeing that first "A>", but it's not just because I'm older and wiser. There are plenty of other significant events around to compete for one's attention.

## Description

The changes made to version 3 are neither unexpected nor original - some of them have been available for some time in various CP/M 2.2 implementations and utilities. But they are nonetheless very welcome and do remove some serious limitations of CP/M 2.2. My remarks here are based upon a pre-release of version 3, so there may be some features which will be changed or unavailable in the release version.

Features of version 3:

### BDOS

- CP/M 2.2 compatibility. (See below)
- 32M/file and 512M/drive maximum size.
- Time and date stamping, password protection.
- Multiple sector read/write.
- System control block containing information about console width, printer list toggles, printer width, time and date accessible to applications programs.
- Direct BIOS calls supported through a BDOS call.
- BDOS disk free function call.
- Blocking and deblocking in the BDOS.
- Physical Disk error flags returned to applications program.
- Automatic log-in of changed disks.

### CCP

- Multiple commands per line.
- Search path for COM files.
- Console I/O redirection.
- Resident System Extensions - Programs which load and stay in memory as extensions of the operating system.

## Comparisons to CP/M 2.2

### COMPATIBILITY

There are some minor restrictions to CP/M 2.2 compatibility. The IOBYTE is, according to the preliminary documentation, going to be dropped, making function calls 7 and 8 up for grabs. IOBYTE has been replaced with a more flexible redirection method, however, by which you can assign any of five logical devices to any combination of twelve physical devices. Function call 27, "Get Allocation Vector", will not be supported. TPA size on computers without bank-switched memory will be reduced by 3-4 K.

### SYSTEM GENERATION

To bring up version 3, you first modify your CP/M 2.2 BIOS. There are one or two code modifications, and an addition of several pages of code. The modifications are not extensive and not all need be done to boot the system. System generation for version 3 is slightly complicated by the fact that all the code is relocatable. If you have written relocatable code before, this is no problem. If not, you may have to change some expressions in your CP/M 2.2 BIOS ASM file listings to avoid expression errors from RMAC.

### CP/M 2.2

- Write BIOS, assemble
- Write Cold Start Loader, assemble
- Write BIOS, CSL, CP/M to System tracks

### CP/M 3.0

- Write BIOS, assemble
- Link BIOS, BDOS
- Run GENCPM
- Write LDRBIOS, assemble
- Link LDRBIOS, CPMLDR
- Write Cold Start loader, assemble
- Write Cold Start Loader, CPMLDR, CCP to system tracks.

### SYSTEM RELOCATION

This is where version 3 is a joy.

### CP/M 2.2

- Re-Assemble BIOS
- Re-Assemble BOOT
- run MOVCPM
- save 43 NEWCPM.COM
- DDT NEWCPM.COM (overlay CSL and BIOS)
- SYSGEN

### CP/M 3

- run GENCPM

GENCPM prompts you for all the information it needs to relocate CPM3.SYS (the file where the BDOS and BIOS resides) for a banked or a non-banked system, then writes a relocated CPM3.SYS.

### BOOT PROCEDURE 0

The "v" in the diagram means "the module above loads and jumps to the module below"

### CP/M 2.2

- Boot ROM
- v
- Cold Start Loader
- v
- BIOS
- v
- BDOS, CCP

### CP/M 3

- Boot ROM
- v
- Cold Start Loader
- v
- CPMLDR
- v
- CPM3.SYS (BDOS, BIOS)
- v
- CCP

CP/M 2.2 (in most implementations) expects all the components of itself to be resident on the system tracks. In version 3, CPM3.SYS is always in a file, CCP and CPMLDR may be either in files or on the system tracks. This allows the two portions of the system which tend to grow in size (BIOS and CCP) to do so freely, and may allow a reduction in the number of system tracks.

If CPMLDR is linked to run at address 100h, it may be run under CP/M 2.2 to boot version 3.

In systems with bank switched memory, a copy of the CCP can be kept in memory and copied down to its load address (100h in version 3) on a warm boot, avoiding the necessity of disk access.

*Next month:* Bringing up a version 3 CP/M.

Version number 2.35 applies to both the full system ("DBASE.COM") and the run-time package ("DBCOD.COM", "DBRUN.COM", "DBRUNOVR.COM"). This version no longer runs under CP/M 1.4.

The run-time package is intended for developers of applications. The application is developed and debugged with the full system. The "CMD" file is then renamed to a "SRC" extension, and "DBCOD" is run on it. A smaller "CMD" file is generated, containing only control character tokens, and can be run with "DBRUN.COM". Only "DBRUN.COM (20K)", "DBRUNOVR.COM (40K)", "DBASEMSG.TXT (12K)", and optionally "INSTALL.COM (12K)" need be present on the applications disk, in addition to the final "CMD" file and any files used by it. For the first time, there is documentation on the file structure of the database included.

The full system is organized differently than the previous version, and contains only three files ("DBASE.COM", "DBASEOVR.COM", "DBASEMSG.TXT"). All the overlay "OVR" files have been gathered into "DBASEOVR.COM", which is 40K long. All messages, including the command descriptions called up with the new "HELP" command, are contained in "DBASEMSG.TXT", an ASCII file which can be edited and lengthened.

There are some new commands:

**HELP <command>** Displays description of <command> syntax and usage.

**TEXT... ENDTEXT** Displays the text between TEXT and ENDTEXT

**REINDEX** Reindexes the database in use to the same index files with which it was opened

**LOAD <filename>** Loads an INTEL HEX file into memory

Listed below are some of the more than 35 changes (i.e., bug fixes) and 26 enhancements documented for this version:

### Changes-Bug fixes

1. Fixed EOF function after PACK of empty index file
2. Fixed INSERT when no data was entered
3. RENAME discards all knowledge of old name after successful rename
4. Fixed problem with LOOP jumping to wrong ENDDO
5. SET LINKAGE ON works with mixture of indexed and non-indexed databases
6. APPEND to empty database shows no garbage with SET CARRY ON

### Enhancements

1. EDIT, APPEND, CREATE, INSERT can use format files to format screen with SET FORMAT TO <file>
2. RESTORE can load saved memory variables without

releasing current memory values by means of ADDITIVE phrase e.g. RESTORE FROM MEMFILE ADDITIVE

3. RELEASE and SAVE can act on subsets of memory variables. e.g. RELEASE ALL LIKE MEM\*
4. DISPLAY STATUS shows USE database file names, indexes, and their key expressions as well as the status of all SETs
5. Relative screen addressing is implemented e.g. @ \$+1, \$+2 SAY "hello"
6. SET DELETE ON causes FIND not to find records marked for deletion.

### Compatibility


1. "MEM" files created under previous versions must be re-created in order to be used with this version.
2. "DBASE.COM" in the full system is 1/4 K larger than the 2.31 version. This may prevent some applications from running if they need that extra 256 bytes for a "SORT".
3. This version will not run under CP/M 1.4.

The remarks below are based upon a beta-test version, and may not be true of the final 2.35 release version.

### Bugs/Oddities

1. Despite a statement in the accompanying documentation that "DO CASE" statements now nest, I found that they did not.
2. Several conditions can cause "DBCOD" to overlook the end of file, and try to generate an infinitely large "CMD" file:
  - a. High-bit characters in the "SRC" file.
  - b. No CRLF after the last statement in the "SRC" file.
  - c. A "TEXT...ENDTEXT" block of more than about 128 characters.
3. A truly odd side effect in using "Select" with "Replace". In the loop below, the "Replace"s will not take effect, even though they are reported if "TALK" is on.

```
use db1
select secondary
use db2 index db2
do while .not. eof
  Store P.ONE to S
  Select secondary
  find &s
  if ##0
    replace P.two with S.two
  endif
  select primary
  skip
enddo
```

The fix is simple, even if illogical. Select the primary database just before the "Replace". For some reason, this works! 

# BOY, IS THIS COSTING YOU.

It's really quite basic: time is money.

And BASIC takes a lot more time and costs a lot more money than it should every time you write a new business software package.

Especially when you could speed things up with dBASE II.

## **dBASE II is a complete applications development package.**

Users tell us they've cut the amount of code they write by up to 80% with dBASE II.

Because dBASE II is the high performance relational database management system for micros.

Database and file handling operations are done automatically, so you don't get involved with sets, lists, pointers, or even opening and closing of files.

Instead, you write your code in concepts.

And solve your customers' problems faster and for a lot less than with BASIC (or FORTRAN, COBOL or PL/I).

## **dBASE II uses English-like commands.**

dBASE II uses a structured language to put you in full control of your data handling operations.

It has screen handling facilities for setting up input and output forms.

It has a built-in query facility, including multi-key and sub-field searches, so you can DISPLAY some or all of the data for any conditions you want to apply.

You can UPDATE, MODIFY and REPLACE entire databases or individual characters.

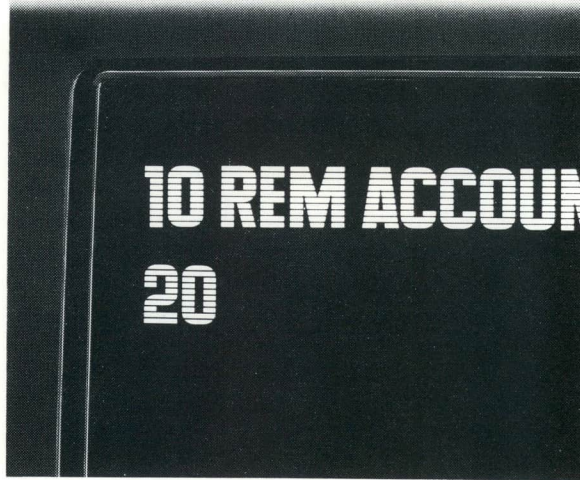
CREATE new databases in minutes, or JOIN databases that already exist.

APPEND new data almost instantly, whether the file has 10 records or tens of thousands.

SORT the data on as many keys as you want. Or INDEX it instead, then FIND whatever you're looking for in seconds, even using floppies.

Organize months worth of data in minutes with the built-in REPORT. Or control every row and column on your CRT and your printer, to format input and output exactly the way you want it.

You can do automatic calculations on fields,



records and entire databases with a few keystrokes, with accuracy to 10 places.

Change your data or your entire database structure without re-entering all your data.

And after you're finished, you can protect all that elegant code with our run-time compiler.

## **Expand your clientbase with dBASE II.**

With dBASE II, you'll write programs a lot faster and a lot more efficiently. You'll be able to write more programs for more clients. Even take on the smaller jobs that were out of the economic question before. Those nice little foot-in-the-data-base assignments that grow into bigger and better bottom lines.

## **Your competitors know of this offer.**

The price of dBASE II is \$700 but you can try it free for 30 days.

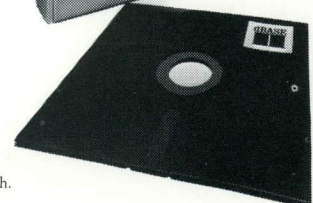
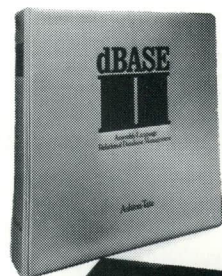
Call for our Dealer Plan and OEM run-time package prices, then take us up on our money-back guarantee. Send us your check and we'll send you a copy of dBASE II that you can exercise on your CP/M<sup>®</sup> system any way you want for 30 days.

Then send dBASE II back and we'll return all of your money, no questions asked.

During that 30 days, you can find out exactly how much dBASE II can save you, and how much more it lets you do.

But it's only fair to warn you: business programmers don't go back to BASIC's.

Ashton-Tate, 9929 Jefferson, Los Angeles, CA 90230. (213) 204-5570.



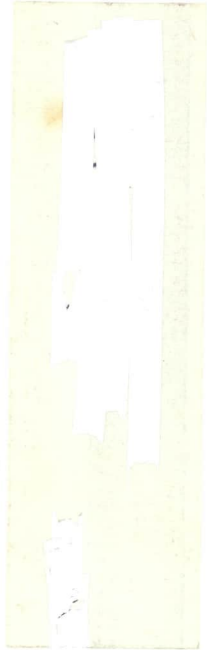
# Ashton-Tate

©Ashton-Tate 1981

**Also available from Lifeboat Associates.**

®CP/M is a registered trademark of Digital Research.

**LIFELINES™** / The Software Magazine™  
1651 Third Avenue, New York, New York 10028



Second Class Postage Paid  
At New York, N.Y.